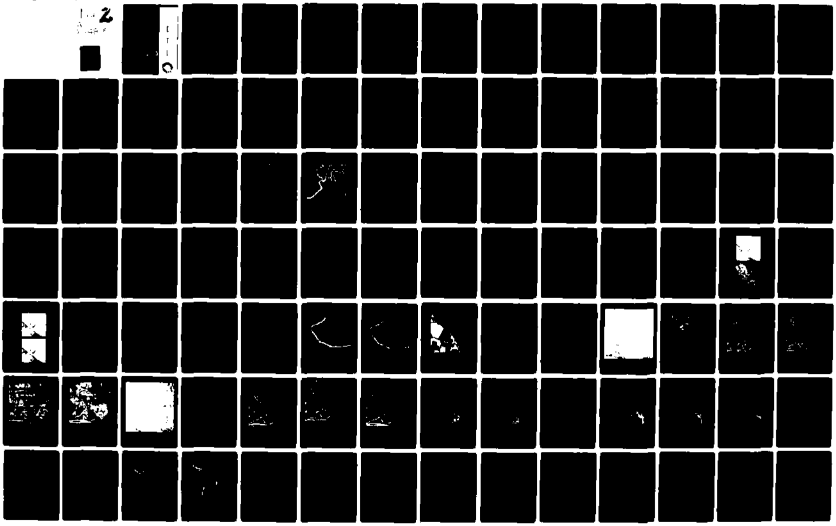
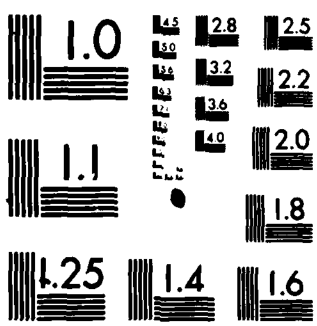
AD-A114 815 GENERAL ELECTRIC CO LANHAM MD SPACE SYSTEMS DIV F/G 9/2
INTERACTIVE DIGITAL IMAGE PROCESSING FOR TERRAIN DATA EXTRACTIO--ETC(U)
SEP 81 T F WESCOTT, W C DALLAM, C J PETERSON DAAK70-79-C-0153

UNCLASSIFIED ETL-0277 NL

1 of 2



1148



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(2)

ETL-0277

Interactive digital image processing for terrain data extraction, phase 2

Thomas F. Wescott
William C. Dallam
Christopher J. Peterson

General Electric Company
Space Systems Division
4701 Forbes Blvd.
Lanham, MD 20706

SEPTEMBER 1981

Prepared for

U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060

E T L

DISCLAIMER NOTICE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ETL-0277	2. GOVT ACCESSION NO. AD-A114 818	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) INTERACTIVE DIGITAL IMAGE PROCESSING FOR TERRAIN DATA EXTRACTION, PHASE 2		5. TYPE OF REPORT & PERIOD COVERED Final Report for Period Sept 1980 - Sept 1981 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Thomas F. Wescott, William C. Dallam, and Christopher J. Peterson		8. CONTRACT OR GRANT NUMBER(s) DAAK70-79-C-0153
9. PERFORMING ORGANIZATION NAME AND ADDRESS General Electric Company Space Systems Division 4701 Forbes Blvd. Lanham, MD 20706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62707A855C0006
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Engineer Topographic Laboratories Fort Belvoir, VA 22060		12. REPORT DATE Sept 1981 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image, Image Processing, Information Extraction, Interactive Analysis, Terrain Analysis, Water Resources Data Element Extraction

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Phase 2 of an experimental study has been performed to investigate the feasibility and use of man-machine interactive digital image processing techniques when applied to the extraction of terrain analysis data from aerial imagery. This phase of the study concentrated on applying techniques to extract water resources data elements and on the development and coding of software and algorithms. The developed software and algorithms, pertained to filtering of themes, and image scaling. Two digital processing

techniques to "skeletonize" features, best mapped as lines, were established and tested with limited success. Relative to water resources data elements, success was achieved in the extraction of alignment of watercourses and shore line alignment of watercourses of water bodies. Partial success was achieved with the elements of delineation of wet areas and the identification of terminal points of watercourse segments. Image processing techniques for the extraction of areas subject to flooding and for the measurement of bank heights were not practical with the existing software.

The development/implementation of software, and the processing and analysis of imagery was conducted at the General Electric Digital Image Analysis Laboratory.

PREFACE

This document is generated under Contract DAAK70-79-C-0153 for the U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia 22060 by the General Electric Company, Space Division, Beltsville, Maryland. The Contracting Officer's Representative was Mr. Joseph H. Kitrosser.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (CONT.)

LIST OF ILLUSTRATIONS (CONT.)

LIST OF ILLUSTRATIONS (CONT.)

I. INTRODUCTION

A. Background

During the period of September 1979 through September 1980, Phase 1 of the experimental study, "Interactive Digital Image Processing for Terrain Data Extraction", was performed. The 12-month study investigated the feasibility and utility of man-machine interactive digital processing techniques when applied to the extraction of terrain analysis data from aerial imagery. The study focused primarily on the extraction of vegetation data elements from digitized panchromatic photography, with a small amount of attention given to thermal infrared and side-looking radar imagery.

Of thirteen vegetation data elements listed in the USAETL Terrain Analysis Procedural Guide for Vegetation, eight were addressed in varying degrees of depth. Interactive digital techniques were developed for vegetation/land cover boundary extraction and for the extraction of several forest-related data elements. For the forest-related elements, three digital themes were developed from digitized panchromatic photography. The three digital themes and a developed automatic operation allowed estimates of percent canopy closure, stems per hectare; crown diameter; stem diameter; stems per hectare per diameter class; and stem spacing to be produced. Best results were achieved with a spatial resolution of approximately three feet in the input digital image.

In addition, techniques were developed enabling the extraction of vegetation/land cover boundaries. These techniques were most effective when the spatial resolution of the digital image is in the range of 8 to 20 feet. With a moderate amount of interactive analysis, processing can be employed to extend the boundary extraction rapidly and efficiently to relatively large areas. Procedures were also developed for distance, elevation, and area measurement from digital imagery. Further testing and development of new technqiues were recommended as a result of Phase 1 work.

B. Objective

The objective of the investigation was to continue the study of man-machine digital image processing techniques to the extraction of terrain analysis data from aerial imagery.

C. Scope

The scope of the second phase part of the study involved the extraction of data identified in the Terrain Analysis Procedural Guide for Drainage and Water Resources. Within the scope was the development and coding of software and algorithms for filtering themes and image scaling. Input data for the second phase included panchromatic, thermal infrared and side looking synthetic aperature radar imagery.

D. Water Resources Elements

In this second phase, the water resources elements considered were:

1. Alignment of watercourses
2. Shore alignment of waterbodies
3. Delineation of wet areas
4. Identification of areas subject to flooding
5. Identification of terminal points of watercourse segments
6. Measurement of bank heights

II. IMAGERY USED

A. Data Selection

Data types used for Phase 2 consisted of two forms, i.e. digital and imagery. Imagery selected as transparencies were digitized into 512 x 512 pixel images (subscenes) with 8-bit gray level quantization. Certain data sets used in phase 1 were applied in Phase 2, specifically pertaining to Fort Belvoir/Woodridge, Va. area and Daedalus scanner images of Pennsylvania. Additional panchromatic photography, synthetic aperture radar (SAR), thermal infrared (TIR) and infrared (IR) data were acquired from the Tennessee Valley Authority (purchased) and General Electric Space Division (loaned) at Valley Forge. Selection of data used for analysis considered the following factors:

- Scene content (various watercourses and water bodies)
- Resolution (watercourses of 1-15 meters width)
- Absence of clouds, cloud shadows and haze
- Uniformity of scene feature brightness levels across the image
- Near-vertical view of the scene
- General quality of the transparency (i.e. absence of scratches, etc)

B. Data Sets

A list of all data sets used for the generation of selective 512 x 512 pixels images (subscenes) is shown in Table 1.

Table 1

1. Elizabeth City, North Carolina; imagery; USAETL
    a. Knobbs Creek Area
        (1) 98^{o} synthetic aperture radar #1 imagery
        (2) 52^{o} synthetic aperture radar #9 imagery
        (3) Panchromatic #39 imagery
        (4) High resolution thermal flightlines 2, 3 and 4

Table 1 (Cont.)

b. Little River Area
   (1) 98^{o} synthetic aperture radar #5 imagery
   (2) 98^{o} synthetic aperture radar #9 imagery
   (3) 52^{o} synthetic aperture radar #5 imagery
   (4) 52^{o} synthetic aperture radar #10 imagery
   (5) 278^{o} synthetic aperture radar #5 imagery
   (6) High resolution thermal infrared #9 imagery
   (7) Low resolution thermal infrared imagery

c. New Begun Creek Area
   (1) 98^{o} synthetic aperature radar #10 imagery
   (2) 278^{o} synthetic aperature radar #6 imagery
   (3) Panchromatic #95 imagery
   (4) Low resolution thermal infrared imagery

d. Chapel Creek Area
   (1) 278^{o} synthetic aperature radar #14 imagery
   (2) Panchromatic #91 imagery
   (3) Panchromatic #120 imagery

e. Symond Creek Area
   (1) Panchromatic #47 imagery
   (2) High resolution thermal infrared imagery

2. Ripley, West Virginia Area, digital, General Electric Company

   a. X-band horizontal polarization
   b. X-band vertical polarization
   c. L-band horizontal polarization
   d. L-band vertical polarization

3. Clarion County, Pa., digital, from Phase 1 study

   a. 20,000 ft. altitude; green and thermal infrared channels
   b. 10,000 ft. altitude; green and thermal infrared channels
   c. 5,000 ft. altitude; green and thermal infrared channels

Table 1 (Cont.)

4. Fort Belvoir/Woodbridge, Va. Area, imagery, USAETL, from Phase 1 study

   a. Panchromatic photographs

5. Tennessee Valley Authority

   a. Holston River set

      - Thermal infrared (TIR) low resolution (flight line)
      - Five panchromatic photographs of flight line area

   b. Night TIR (low air temperature)

      - Six panchromatic photographs

   c. Reservoir

      - TIR
        Five panchromatic photographs

III. TECHNIQUES DEVELOPMENT

A. Image Resolution and Scale Changing

As reported in the first phase of this project, spatial operators, which include the reported microtexture and crown template operators, are very useful tools for the interpretation of monochromatic imagery. The program of successfully categorizing monochrome imagery is in general a much more difficult program than with other types, such as multispectral, because of the relatively smaller amount of information available. Attempting to categorize an image by level slicing alone encounters serious problems, the most difficult being that signatures tend not to be unique enough to be used as the sole descriptor of the classes generally desired. Even seemingly simple classes like water bodies can be impossible to extract via level slicing. Clearly more information than gray level is required to perform categorization, and clearly such information is available, in the form of spatial patterns.

In particular, textural information is of considerable value in the analysis of high resolution images. Texture is an encompassing term with a multitude of types and definitions, most of which have limited use in remote sensing. The most basic and perhaps most widely applicable type of texture is simply surface roughness, which can be relatively easily determined by computing local image variance. Textural operations are scale dependent and for effective use, must be applied at a specific scale which is defined by the size of the microfeatures which generate the rough appearance of larger features being extracted. For example, a typical image might contain three features which we wish to separate, pasture, brush land, and forest. Each of the features is basically a random mixture of vegetation and shadow, and quite likely all three exhibit approximately the same integrated reflectance. Each of the features also exhibit similar textural structure, the appearance of random roughness, but they are distinctly different in textural scale.

To realize the full potential of scale dependent spatial techniques we must be able to efficiently vary the size of the spatial operators used (or at least the effective size), allowing the extraction of a

considerable range of spatial features. This scaling can be accomplished by either of two methods: (1) by scaling the operator itself, which is the historic method, or (2) by changing the scale and resolution of the image and applying a universal operator. The image scaling approach appears to have several significant advantages:

- Scaling of all but the simplest of spatial operators is not as straight forward as is reducing resolution and scaling an image. Any resolution up to that inherent in the original data can easily be simulated with very simple and rapidly executed algorithms. Allowing, for example, the highly effective and reasonably sized max-min microtexture operator to be used without modification, in the extraction of a wide variety of textural identifiable features.

- Matching spatial feature templates to objects of a particular size can be done much more precisely by changing image scale, as finer scaling increments can be used. Many effective spatial operators cover areas as small as 3 x 3 pixels, thus the finest scale adjustment would be a factor of 1/3.

- The reduction in the computation time required to apply an equivalent operation is significant when image scaling is chosen. Reduction in resolution means reductions in the volume of data processed. Reducing image size by a factor S reduces the number of pixels to be considered by S^2, compared with increasing the operator size, which would increase most spatial computations by S^2. The relative difference approaches S^4.

During Phase 1 of this study, image scaling was accomplished by utilizing those techniques which were readily available at General Electric DIAL without software development. While the usefulness of image scaling to expand the utility of standard and simple spatial operators was firmly established, the actual methods were cumbersome and time consuming.

The ability to vary image scale and resolution rapidly enough for use in interactive analysis was quickly identified as being a missing and necessary tool.

Given the task of using spatial information as an input for the categorization of some image feature, a fairly lengthy procedure might be followed. First the image would be reduced in scale and resolution by some factor defined by the predominant spatial frequency of the object to be extracted. For example, the mean distance between shadows in forest canopy. A spatial operator is then applied to the reduced image generating a derived image where the spatial feature has been extracted or enhanced. This secondary image is then enlarged to the scale of the original for subsequent analysis. Since the derived image contains information of a nature independent from the spatial grey level information of the original, this operation essentially creates a multidimensional feature space which significantly increases an analyst's options when attempting to categorize a monochrome image. Repeating the operation at several scales generates an even more interesting feature space of higher dimensionality.

Based on the experience gained during the Phase 1 analysis of several images, a program was written to perform image scaling by any of three operator selected interpolation methods, nearest neighbor, area averaging, and a fairly unique subpixel interpolation method based on local scene context. Figure 1 is a flow chart of the program.

1. Scale Changing via Pixel Replication

Often simple replication of pixels is a satisfactory or even desirable method of changing the scale of an image. Whether or not it should be the chosen method is dependent on the specific image processing task being performed. For example, if one desires to quickly examine an 'overview' of a large image containing too much data for complete display on a CRT, interpolative resampling with its associated computation could be a waste of time. In other cases, such as resampling an image with a very small scale change, no noticable degradation will be suffered with this technique. Finally, there are many applications where original radiometric values must be preserved. In these cases replication or

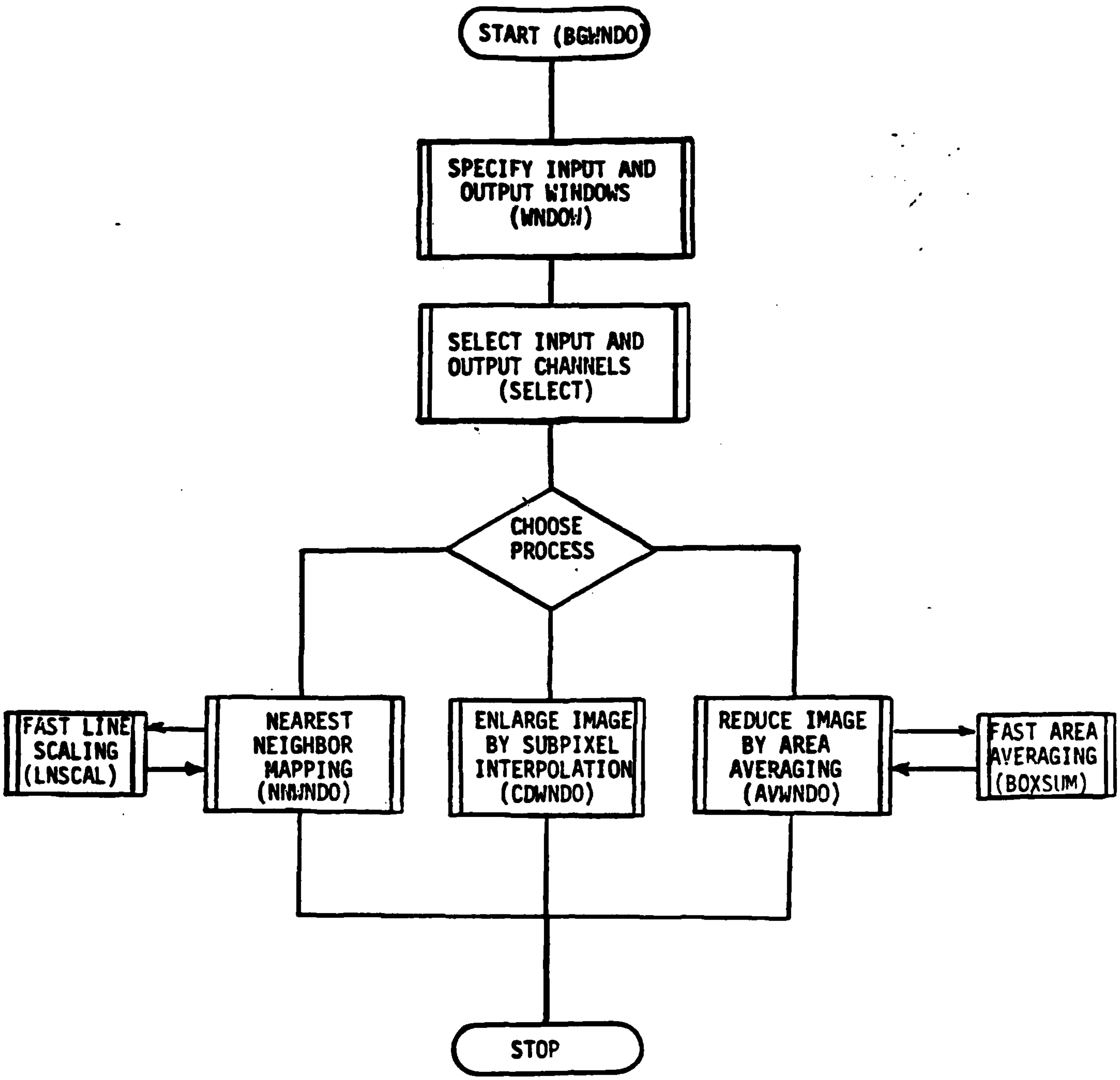


Figure 1. Image Scaling Program Flow Diagram

nearest neighbor resampling is the only simple method. Examples are, when sensor characteristics are being extracted, or when thematic or binary images, which are not gray scale in nature must be scaled.

Pixel replication sounds like a trivial problem, but the capability is so basic and heavily used in image processing, that a serious look at implementation and execution time is worthwhile. In several recent image processing systems the implementation is via hardware and operations are performed in near time. For other systems, a software solution is very reasonable, as rapid execution can be achieved.

Since no real pixel by pixel computation is required, either in determination of output pixel value, or in determination of output position, algorithms can be used which amount only to moving data in a high speed manner from a predetermined location to a predetermined destination. This is accomplished by setting up two address tables based on the replication or sampling required along lines and between lines. These tables, one with an element for each output line and one with an element for each output pixel, contain addresses in the input image. Each of these addresses is then the source location for each desired output pixel.

Using these tables, the actual execution is performed by indirect addressing. Data movement can be done by several methods including special purpose hardware and replacement statements from a high level programming language. In this case replacement was done in an assembler routine to minimize execution time. Omitting I/O limitations, the program takes approximately three seconds to construct a 512 x 512 pixel output image. Figure 2 shows the algorithm used.

2. Fast Area Averaging

This capability is fundamental for the extraction of scaled spatial information from high resolution images in the manner described in Phase 1. Starting with an image digitized and stored at the highest desired resolution area averaging is used to generate a secondary image of "super pixels" where digital numbers are related to the integrated

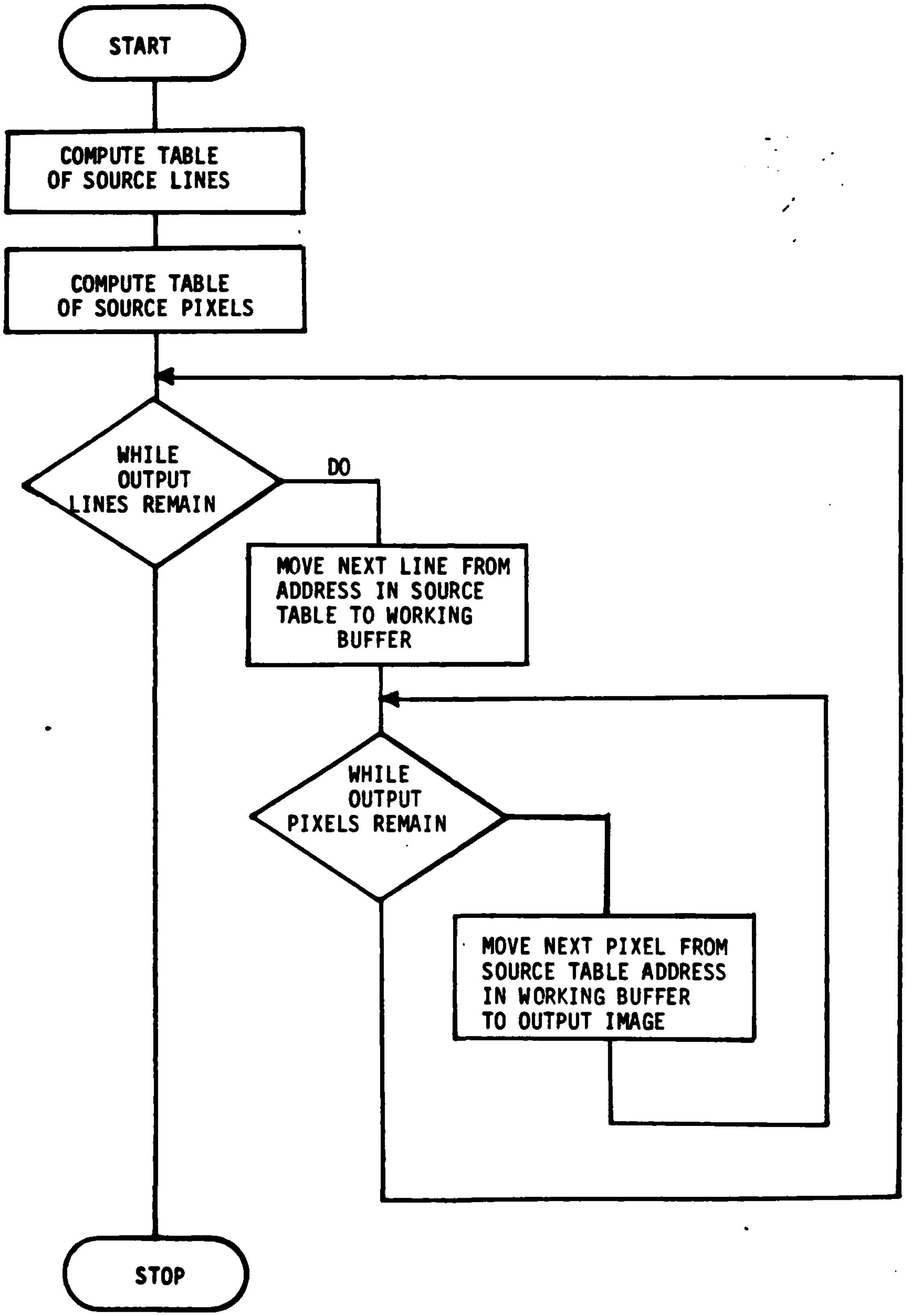


Figure 2. Scale Change by Replication

reflectance from a large number of original pixels. The spatial frequency content of the derived image is cut off by the span of the averaging area, simulating an image that might result if the same sensor and objects were used in a lower resolution (higher sensor altitude) configuration.

In use, as the averaging area increases, the observable texture of finely structured areas disappears from the derived image. Textural information can then be extracted by differencing the derived image with the original in a manner similar to a technique widely used for edge extraction. Area averaging is then applied to the derived difference image as a smoothing function which generates a grey level image where the digital number is proportional to the local high frequency content (texture) cut off by the original application of the filter.

The approach chosen to implement the filter was to use an adaption of the highly efficient box filtering technique. Box filtering computes a moving average over a rectangular area by using two linear moving averages, one down image columns and one along image rows. Execution is very efficient. If the box size is small in relation to the image size, the number of operations required to compute one output pixel approaches four, and is independent of box size. While box filtering computes more output pixels than are needed for scale changing, it is unlikely that any other algorithm can approach this efficiency. The scale changing program computes all of the column averages used in box filtering, but computes row averages only for the required output lines. Required output pixels are stripped from the row average at regular intervals. All along row operations are done in assembler subroutines to minimize execution time.

As with all spatial operations, some trickery must be used to avoid undesirable image border effects without reducing the size of the output image. Since, for our application, the derived images are often to be recombined with their full resolution parents, maintaining image size is very important. This need is particularly acute if large filters are used. Data must, in effect, be manufactured so that the moving averages

do not contain pixel values of zero at image edges. The method chosen as being the least objectionable, was to increase input image size by reflecting data values about image borders. Figure 3 shows the algorithm for fast area averaging.

3. Sub-pixel Interpolation for Image Enlargement

Selection of an interpolation algorithm for use in image enlargement is not quite as straight forward as many recent papers on the subject would lead us to believe. Truncated or otherwise modified versions of the "theoretically perfect" $\left(\frac{\text{sine}\,x}{x}\right)$ interpolation have been used to achieve good interpolation with a minimum of computation. However, some interesting artifact producing properties are made evident when we use these interpolators an an enlarging tool. In fact, making an enlargement is an excellent means of graphically showing many of the problems associated with various algorithms.

The often used and much discussed "cubic convolution," for example, which computed an interpolated value based on 16 surrounding pixels produces a curious effect on edges with certain orientations. When the point to be interpolated falls directly on one of the lines defined by the pixel grid of the original image, this algorithm performs admirably. In these cases, the algorithm degenerates to a one-dimensional operation. When the interpolation falls between the lines, two-dimensional interpolation is performed and considerable aliasing can occur. When the original image contains an edge feature oriented at 45^{0} to the pixel grid, the aliasing is most apparent. The optimum $\left(\frac{\text{sine}\,x}{x}\right)$ interpolation would produce a blurred (from enlargement) but straight edge in the output image. The cubic convolution reproduces the 45^{0} edge with a wobbly sine wave appearance, showing that individual interpolated values are as much a function of resampling position as they are of image content.

The other commonly discussed interpolator, bilinear interpolation, has been shown in numerous studies to perform just a little worse than cubic

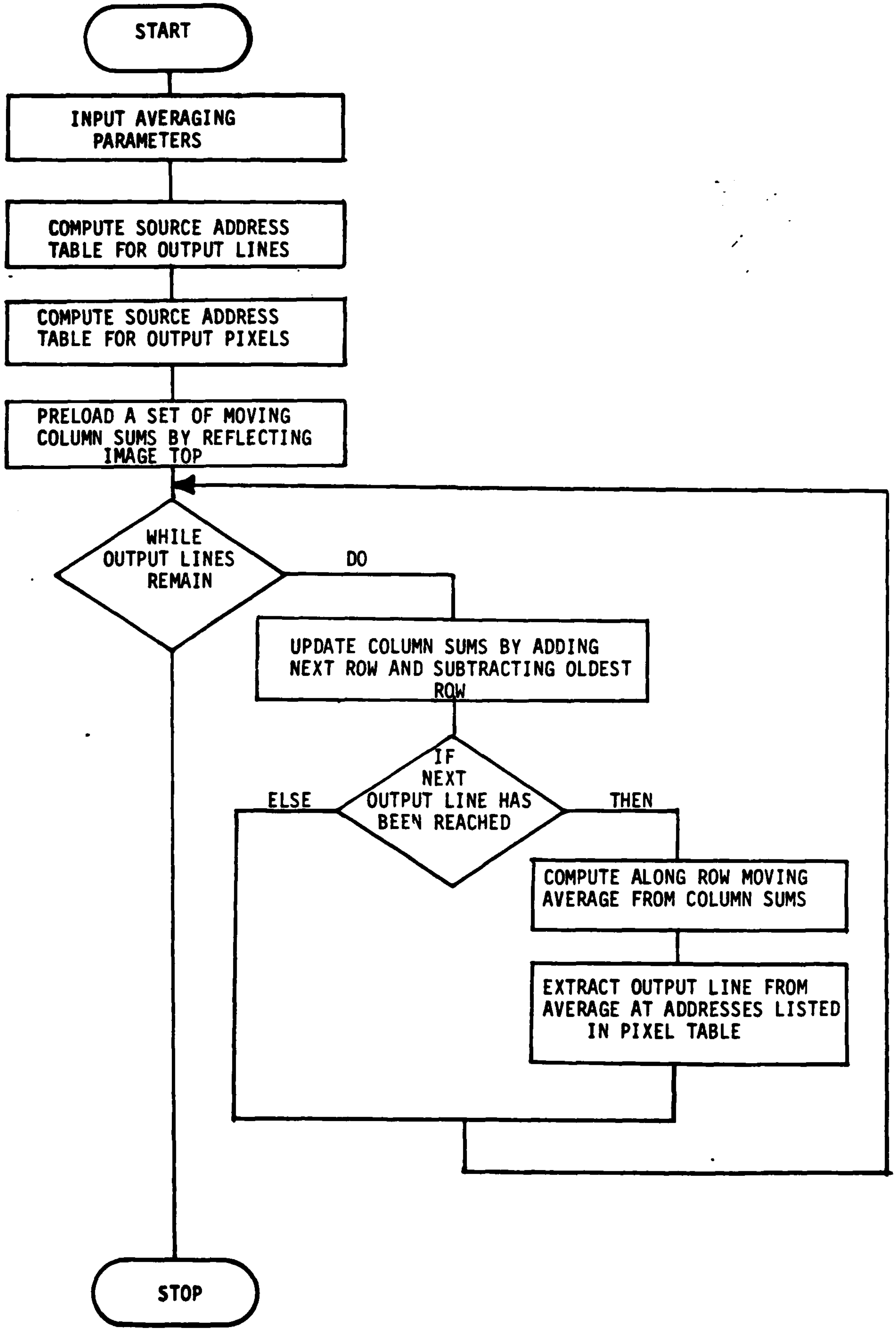
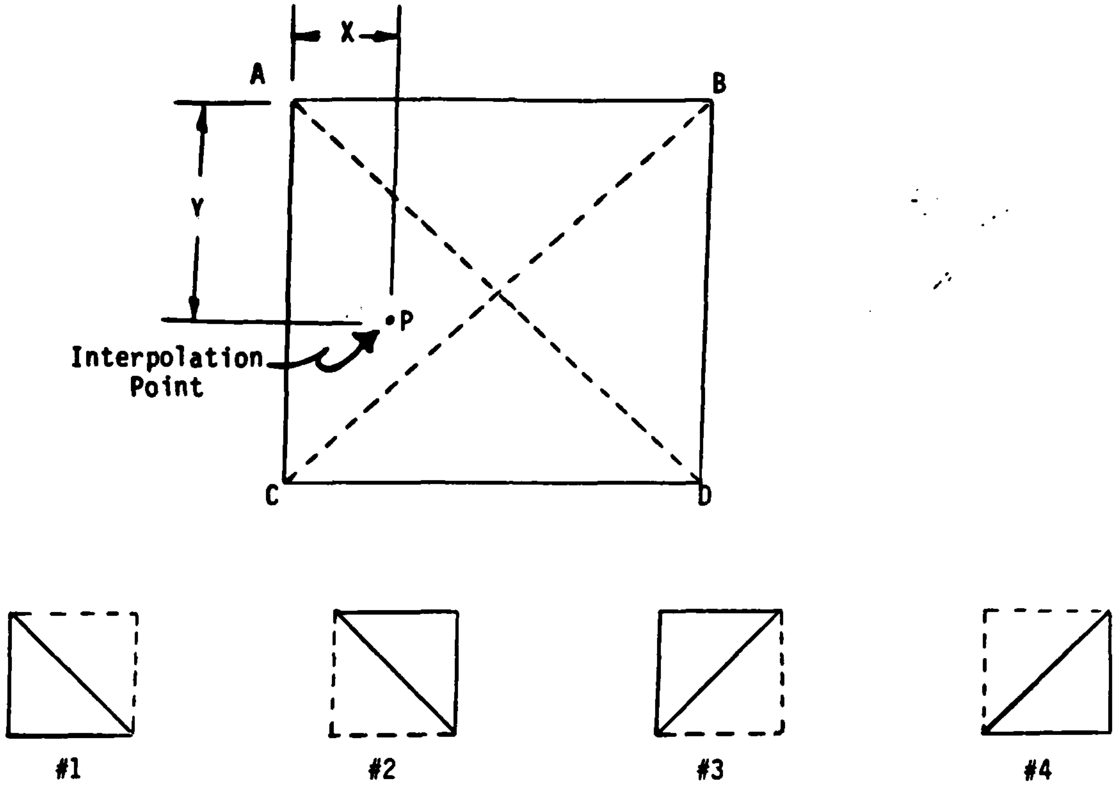


Figure 3. Fast Area Averaging

convolution in just about every respect. True to form, when subjected to a visual "enlargement test" it shows 45° artifacting a little more severe than cubic convolution does. Considering that it only requires four points, the algorithm does not perform all that poorly. Bilinear interpolation basically performs linear interpolation along two opposite sides of a quadralateral defined by four pixels, and then performs a third interpolation between these two new points. It then, is based on the assumption that all four points lie in the same plane, a gross assumption which is incorrect whenever the four points are over an image edge.

The interpolation scheme chosen was a four point interpolation which breaks the quadrilateral defined by the four surrounding points into triangles. The most appropriate traingle for interpolation is chosen by the direction of the local image gradient and by the position of the desired point. The interpolation is then controlled by image content and handles 45° edge situations well. See Figure 4.

The context dependent interpolation has been coded and tested at GE DIAL and performs as was expected. The 45° artifacts are completely eliminated. The interpolator does have some remaining problems which could ultimately show up in factor analysis. Angular artifacts remain at $22\frac{1}{2}^{\circ}$ to the pixel grid and isolated points are reproduced as diamond shapes. Further use in the generation of factor overlays should determine the value of this method. It is likely that some combination of a linear interpolators and a context dependent interpolator would produce better results.



FOUR POINT CONTEXT DEPENDENT INTERPOLATION

STARTING WITH INTENSITY VALUES A,B,C,D SURROUNDING THE DESIRED POINT LOCATION, SELECT FROM THE FOUR DEFINED TRIANGLES, THE ONE INCLUDING BOTH THE POINT LOCATION AND THE DIAGONAL WITH THE SMALLEST INTENSITY GRADIENT. LINEAR INTERPOLATION IS DONE ON THE PLANE DETERMINED BY THE SELECTED TRIANGLE.

```
CASE #1     IF ((|C-B|≥|A-D|) .AND. (Y≥X))
               P=A+X(D-C)+Y(C-A)

CASE #2     IF ((|C-B|≥|A-D|) .AND. (Y<X))
               P=A+X(B-A)+Y(D-B)

CASE #3     IF ((|C-B|<|A-D|) .AND. (X≤(1-Y)))
               P=A+X(B-A)+Y(C-A)

CASE #4     IF ((|C-B|<|A-D|) .AND. (X>(1-Y)))
               P=C-X(D-C)+(1-Y)(B-D)
```

Figure 4

B. Feature Skeletonizing

At the end of Phase 1, another capability seen as very useful for the extraction of several terrain elements was the ability to transform those elements characterized by filament type objects into single pixel width lines. Two obvious examples of elements requiring the operation are the delineation of watercourses and roads. There are really two different levels to the program. The first case, where the feature is spatially connected i.e. no breaks, is relatively easy to handle by using a line stripping algorithm to reduce feature width to the point where only one strip of adjacent pixels remain, bisecting the original feature. In general, however, themes are not always well connected. As the desired feature nears the resolution limit of the imagery, or is hard to categorize for some other reason, the feature is usually mapped as a tenuous string of pixels spaced with many gaps. Any widely applicable skeletonizing algorithm should be able to bridge at least small gaps and then shrink the feature to produce the single pixel width map.

The problems are demonstrated vividly when one attempts to extract watercourse alignment. As described in the element extraction section of this report, several methods of extracting large watercourses are feasible and success was achieved with most of the images investigated. The most widely applicable methods rely on image intensity slicing in combination with the unique smoothness or lack of texture that water bodies exhibit to most sensors. Extracting the alignment of very narrow watercourses is a different problem as there is not enough resolution available to make use of texture or any other positively identifying features. While the watercourses can be roughly identified by density slicing, it is difficult to classify them as contiguous objects. Thin or intermittent watercourses are often revealed to the photo interpreter, not by the presence of clearly identifiable water, but only by subtle changes in the surrounding vegetation, or as a linear shadow with no water being imaged.

Prior to a skeletonizing operation, connectivity of the feature must be established if we are to work successfully with narrow watercourses and roads. Some encouraging results in this area were obtained by performing a very low pass filtering operation (box filtering) on unconnected binary theme maps. Filtering in this manner generates a slowly varying grey level output where losses in connectivity have been "bridged" by the slowly varying intensity ramps. The operation transforms strings of isolated pixels into a series of peaks and cols. Level slicing on the smoothed image results in a map much wider than the original feature, and one where connectivity has been established across gaps of width up to the size of the filter. The method has two undesirable effects. First the width of the feature is increased by up to twice the span of the filter size, making subsequent skeletonizing by line stripping methods a more lengthy process. Second, the filtering introduces a straightening of the feature centerline. The straightening can become pronounced and objectionable when filter size approaches the dimensions of feature characteristics such as stream meanders. Figure 5 shows a sample watercourse theme and Figure 6 is the slice where minor gaps have been bridged.

1. Skeletonizing Features via Gradient Zero Crossing

Several attempts were made to utilize linear spatial operators to locate feature centerlines. The results were less than satisfactory, and many problems remain to be addressed, but the use of differentiating operators on grey level images generated from smoothed themes, does present some interesting possibilities.

A binary watercourse theme is first changed by low pass filtering, into a continuous grey level image with a continuous first derivative. In this smoothed image, the watercourse center is characterized by a local intensity maximum which defines a ridge line curve where the local gradient taken perpendicular to the watercourse direction changes sign as it passes through zero. Zero crossing is a uniquely descriptive feature which is independent of the watercourse width, and also independent of the gray level magnitude corresponding to the watercourse. As long as the smoothing operator has established some non zero grey level

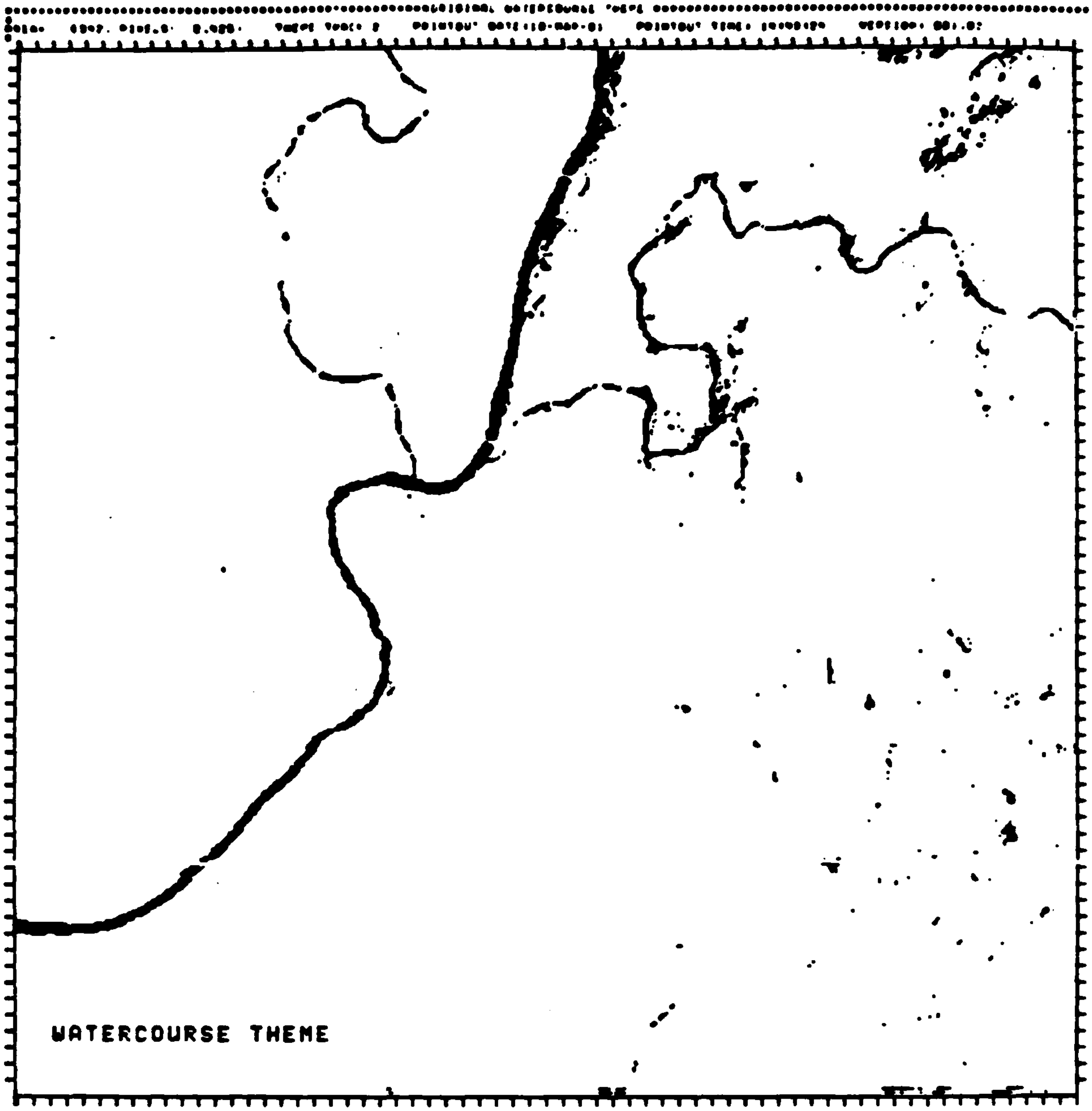


Figure 5. Typical Watercourse Theme (Purposely Left Rough to Provide a Difficult Test)

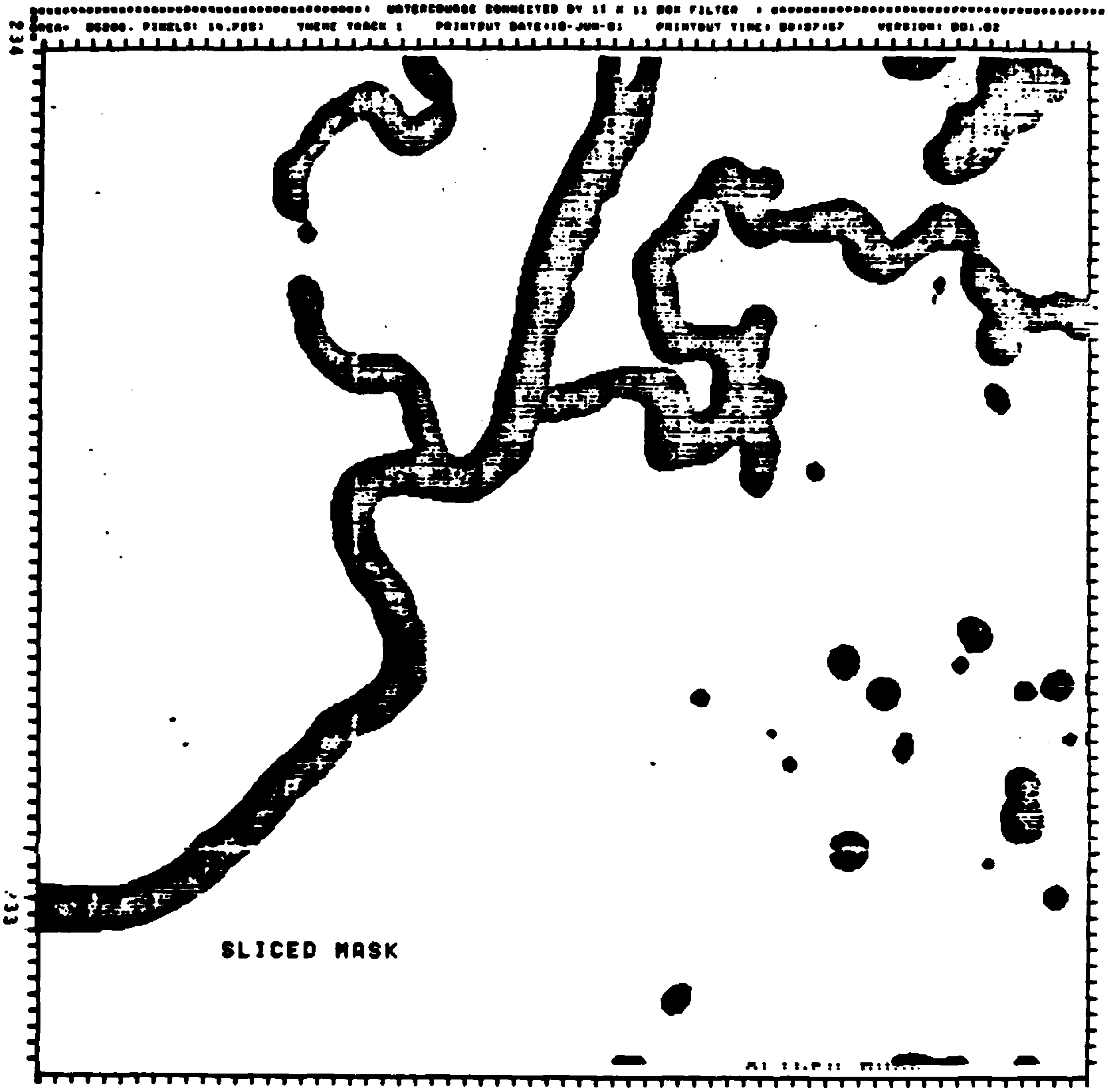


Figure 6. Watercourse Mask Where Connectivity Has Been Established By Slicing A Low Pass Filtered Image.

connection between watercourse pixels, a gradient zero crossing line will exist and can be located. Mapping the zero crossing can be achieved by level slicing all negative (or positive) pixel values and then outlining the area mapped by the slice. Figure 7 is a schematic of the processing sequence.

The zero crossing line has some very attractive features:

- As long as the filter size is greater than one half the watercourse width, gradient zero crossing will occur at the watercourse centerline.
- The zero crossing line tends to establish and maintain connectivity across small gaps in the watercourse.
- The line can be quickly and easily mapped into a single pixel width feature.

Despite the promise of this method, only limited success was achieved when applied to real images. The method was useful on spatially simple watercourses i.e. those without meanders, where the entire stream could be projected in some direction with a one to one mapping, allowing the use of a single gradient direction, as is shown in Figure 8. The method could be used on similar features with limited curvature. Whenever the range of curvature direction was greater than a semicircle, the method was unsuccessful. The problem reduces to one of preserving some of the properties of a direction sensing operator while making the operator work over the image in a directionally invarient manner. There is likely a solution which would make the method universally applicable, but during this project, time limitations brought an end to the investigation.

Most of the obvious adaptations of the gradient method were tried. Taking gradients from two directions and then combining the result into an image where grey scale was related to gradient magnatude failed. In general connectivity was not well preserved, and it was lost at every stream junction point. Likewise, treating orthogonal directions independently failed. This approach produced results which were not totally coincident and thus could not be recombined to form a single centerline.

(1) Original Watercourse Theme

(2) Smoothed Theme

(3) Sliced Smoothed Theme

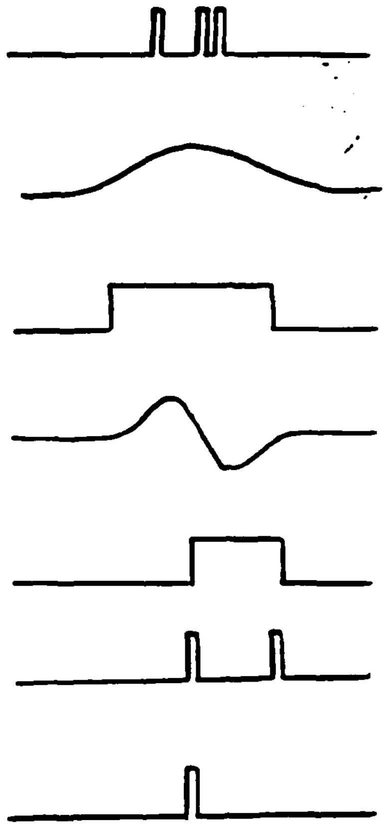
(4) Gradient of Smoothed Theme

(5) Sliced Negative Gradient

(6) Outline Gradient Slice

(7) Logical Combination of of (6) and (3)

(A single pixel width Watercourse alignment map)



Reduction of an unconnected watercourse theme to a single pixel width watercourse alignment map by using low pass filtering and zero crossing of gradient.

Figure 7. Processing Sequence for Skeletonizing

Figure 8. Watercourse Skeletonizing by Gradient On Spatially Simple Stream

The most interesting approach, which also failed, was to use a rotating gradient operator. With a 3 x 3 pixel moving operator four gradients arranged in a semicircle were extracted. Output of the operator was the signed magnitude of the gradient possessing the greatest absolute value. The idea was to always extract the gradient in a direction perpendicular to the watercourse so that the zero crossing line would be preserved, even when the gradient sign was reversed. Unfortunately the operator generated unacceptable discontinuities whenever the watercourse direction was oriented perpendicular to the line defining the semicircle. See Figure 9.

2. Skeletonizing Features via Golay Processing

Golay processing is a purely spatial logical operation which is applied to a binary theme map. For each pixel position, the operation examines pixels immediately adjacent to the central pixel, and adds or deletes the central pixel from the output based on the configuration of its adjacent neighbors. Golay transforms were meant to operate on images arranged in a hexagonal array where each pixel has six neighbors. There are several significant advantages to this arrangement when compared to the more commonly used rectangular array. These advantages are: fewer pixels are needed for spatial operators, between neighbors interpixel distances are constant, and a higher order of symmetry exists. Few interactive systems, however, work on hexagonal arrays, making implementation on these systems complex. At CE DIAL, the transforms have been applied in a compromising manner directly to a rectangular array, resulting in a capable program but also one which has some directional artifacts. Figure 10 shows the Golay patterns as implemented at DIAL.

Given a connected watercourse theme, Golay processing is a reasonable means of shrinking the watercourse to the single pixel width line needed to identify alignment. Skeletonizing is accomplished by deleting from the watercourse theme, all pixels with surrounds numbers three, four, and five as labeled in Figure 10. The operation basically removes the outside annular ring of pixels from every contiguous area and thus narrows the feature by two pixels per pass.

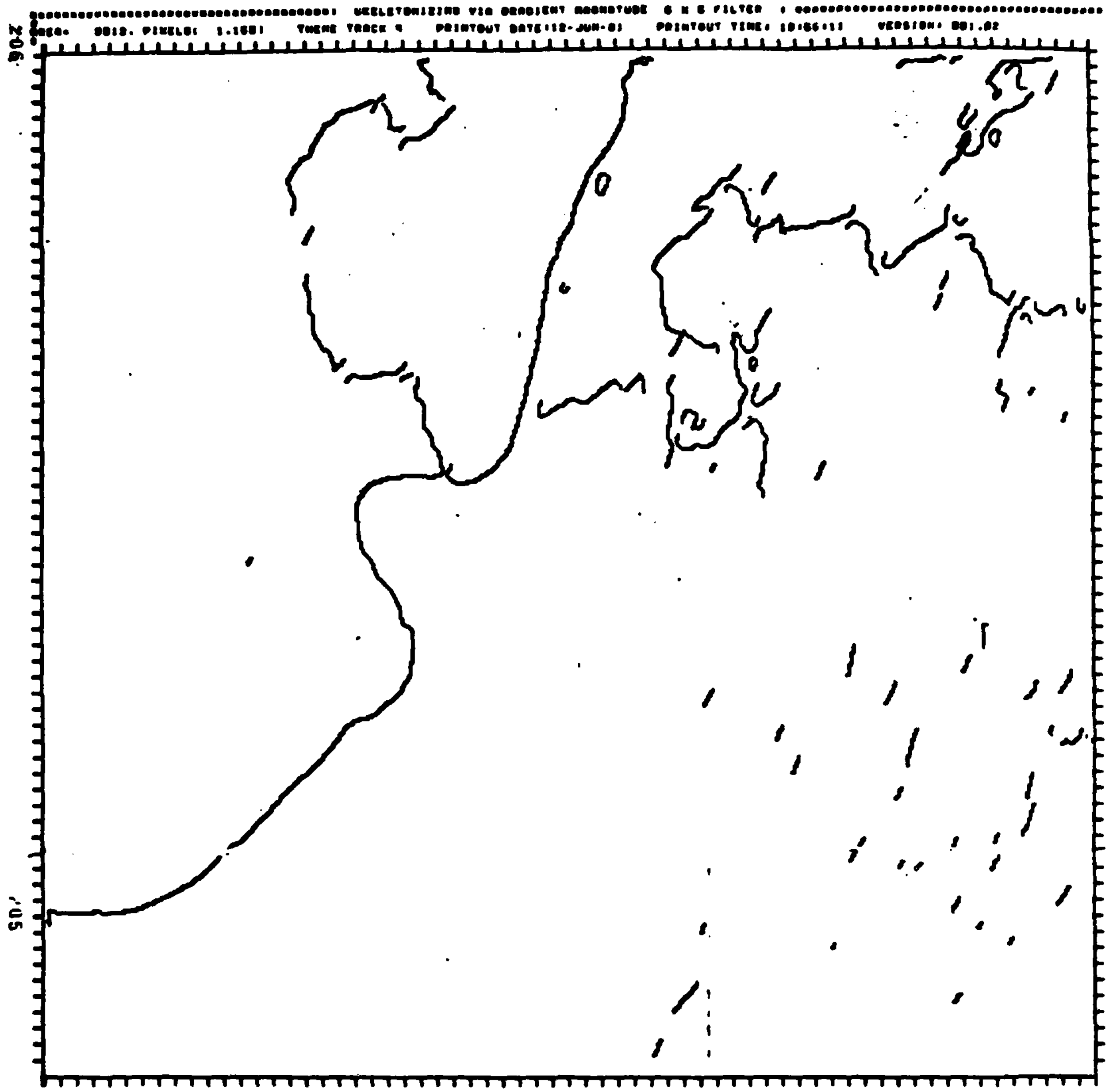
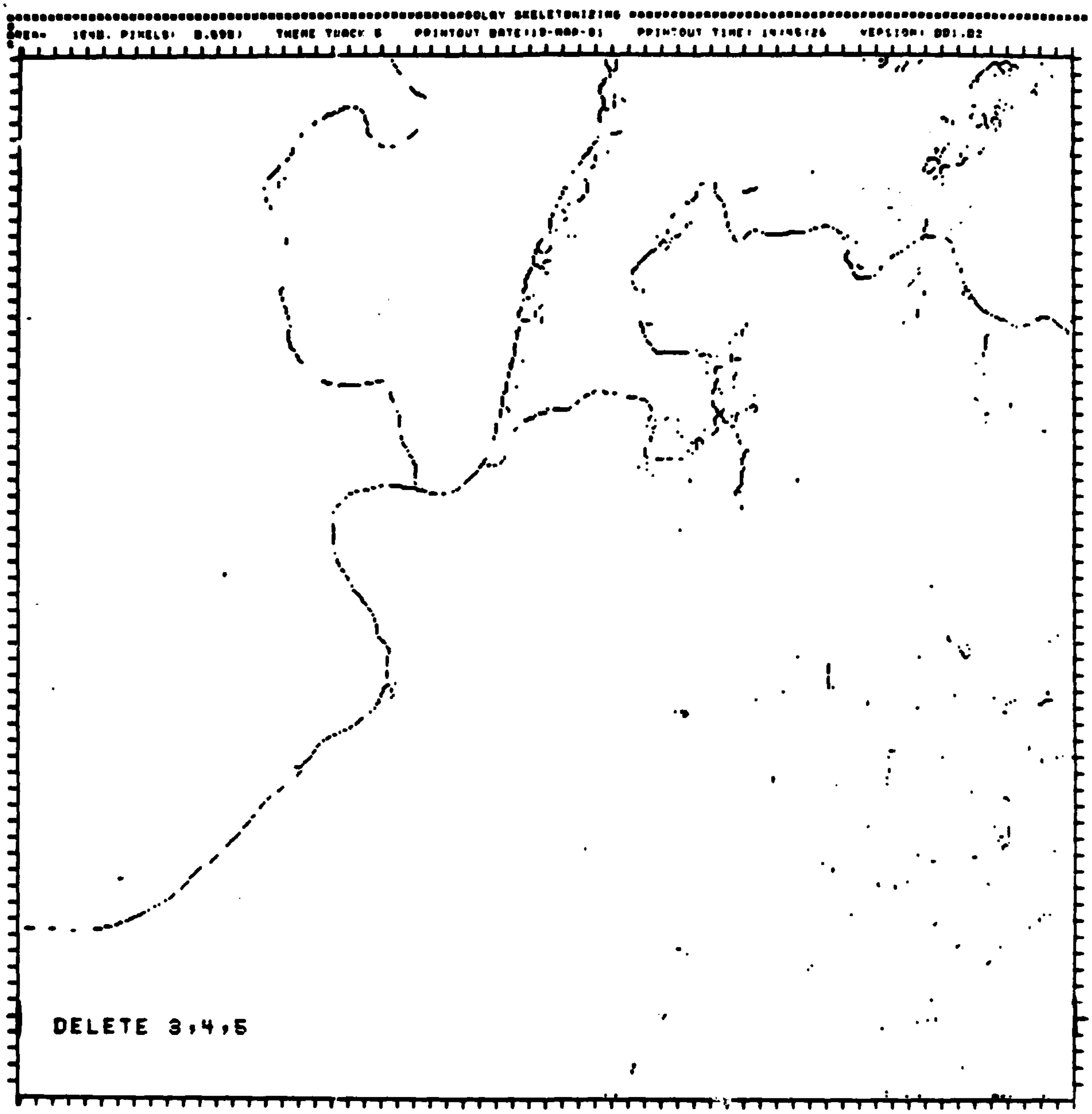


Figure 9. Unsuccessful Attempt At Watercourse Delineation By Zero Crossing Line Of Maximum Local Gradient. Note That Discontinuities Are Introduced Only Then Watercourse Is Oriented At 10 o'clock.



THE 14 GOLAY SURROUNDINGS ARE BY NUMBER:

```
00     10     11     11     11     11     11
0X0    0X0    0X0    0X1    0X1    0X1    1X1
 00     00     00     00     01     11     11

#1     #2     #3     #4     #5     #6     #7


10     11     11     11     10     00     11
0X1    0X0    0X0    0X1    0X1    1X1    0X0
 10     01     10     10     00     00     11

#8     #9     #10    #11    #12    #13    #14
```

Figure 10. Golay Skeletonizing of Watercourse Theme

A problem with the method results from the small neighborhood sampled by the Golay surround, this or any other rotationally invarient operator of 3 x 3 pixels or less, cannot sense the difference between the boundary of a large contiguous area, and a narrow linear feature two pixels wide. The result can be deletion of pixels from both sides of the narrow feature and thus loss of connectivity. Figure 10 shows the same watercourse skeletonized in this manner. There are ways to stop this action. For example, slight smoothing of the feature, a technique which has already been recommended to establish connectivity, can also be used to introduce curvature into long straight boundaries minimizing occurrance of the malfunction. Other techniques, such as slightly randomizing feature boundaries have also been used.

Alternatively, a modification of the process avoids the problem, but introduces other effects. The inability to recognize lines of two pixels width, is avoided if the process is changed so that image raster lines being processed are subjected to immediate updating. With updating, whenever an output pixel is changed, the surround of the subsequent pixel includes that change. The compromise with this approach is that the resulting single pixel width line is no longer centered on the feature. The line is instead shifted to the side of the feature which was last processed. Even with this undesirable effect, the technique might be of value for use on narrow watercourses where the induced offset is small. One very attractive feature of this modification is that two passes over the image are usually sufficient which can mean a significant reduction of execution time. Figure 11 shows a sample result.

Based on the approaches to skeletonizing investigated, the use of Golay pattern transforms of similar operations is viewed as the more promising means to achieve the delineation of narrow watercourses. The properties of rotational invariance, moderate execution time, and a versatile selection of operations make it useful. The main deficiency of the approach is that watercourse gaps of more than one pixel cannot be easily bridged, but this can be overcome by filtering. Figure 12

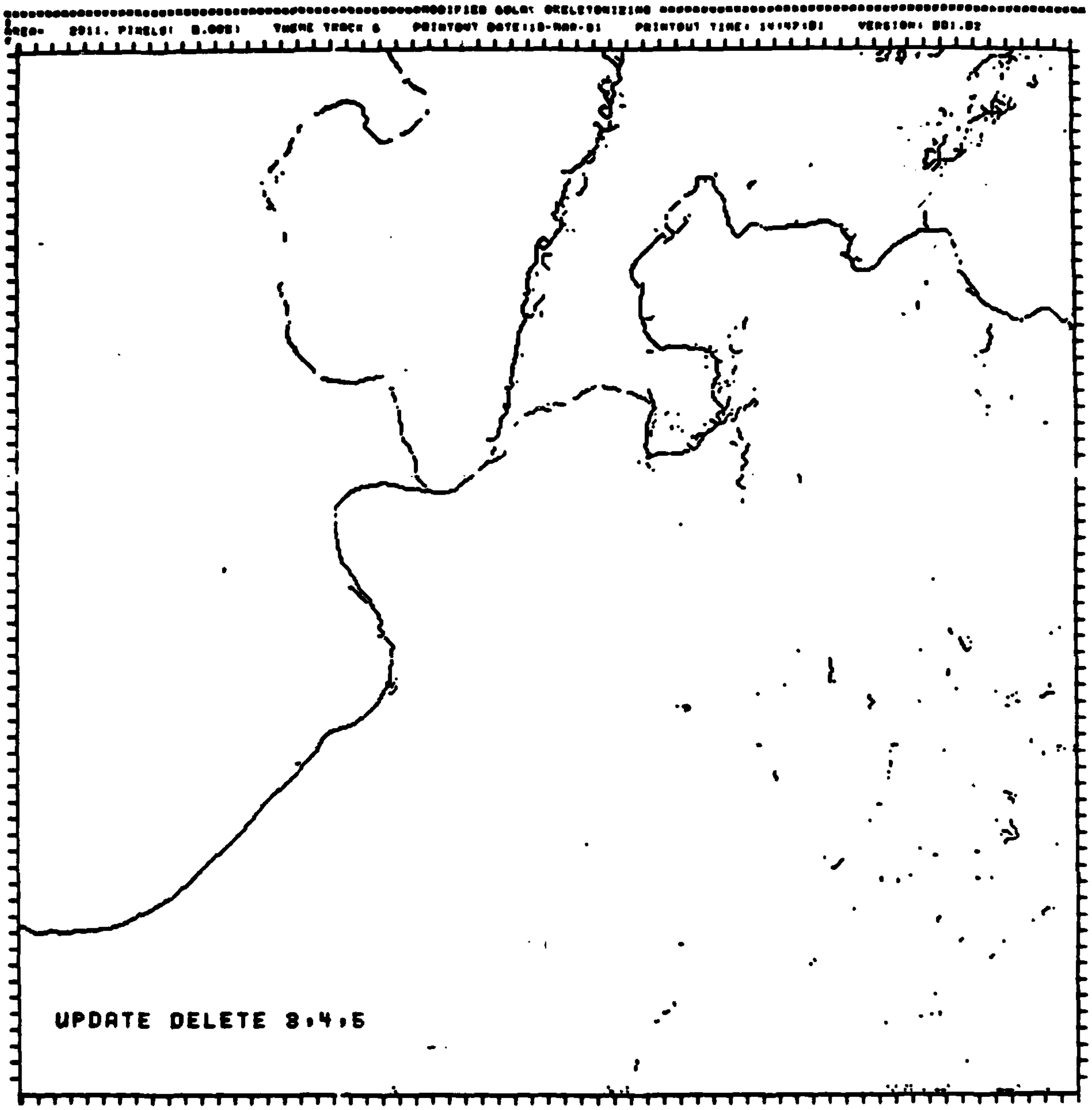


Figure 11. Golay Skeletonizing With Updated Image

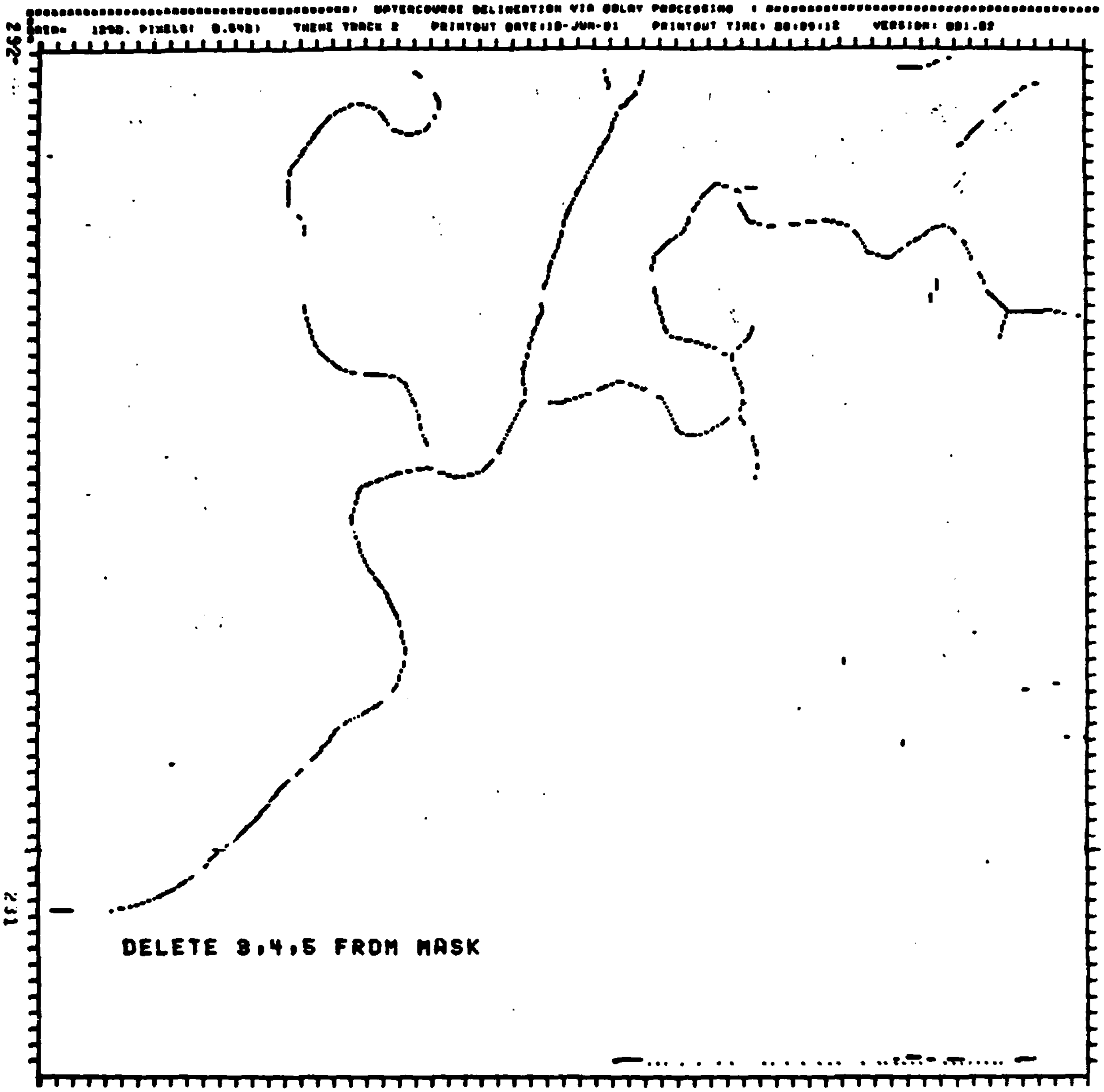


Figure 12. Watercourse Delineation by Skeletonizing the Mask In Figure 6.

represents Golay skeletonizing of the smoothed theme, and is probably the most successful skeletonization exercise. The gaps in this figure are largely a result of the way Golay transforms are currently implemented in GE DIAL with no attempt to compensate for use on a square pixel grid. These deficiencies could most likely be removed with an improved program.

C. Theme Smoothing Method

Categorization of images by computer assisted methods produces, except for the extraction of very easy and distinctly unique classes, results that are spatially complex. The classification may or may not be "noisy", depending on the suitability of the data for extraction of a given class, but will in general exhibit a salt and pepper appearance which might be interpreted as noise. (This effect comes from the fact that with current technology computers perform relatively poorly at spatial tasks.) Most of the commonly used classification schemes are totally aspatial, relying on intensity level in one or more spectral wavelenth bands, while completely ignoring feature boundaires, shapes, and texture. Even those methods which aspire to use spatial information, for example, the textural operators described in this report, use spatial information in a very rudimentary manner when compared to the feature extraction capabilities of a human image interpreter.

The interpreter, in contrast, has a relatively poor ability to identify image intensity levels with any real accuracy, and relies very heavily on his spatial abilities, particularly when analysing monochrome imagery. As opposed to a computer classification, which takes place on a per pixel basis, the interpreter performs classification on a per feature basis. This leads to extraction of features by the location of their external boundaries and tends to force the classes into contiguous areas. The interpreter tends to ignore (or accomodate for) minor variances internal to the feature which might affect a computerized classification, and also performs considerable smoothing or straightening of feature outlines.

Deciding which of these two radically different approaches is more accurate is moot, and depends on ones definitions and objectives. What is certain is the fact that the interpreter generally produces a map that is far more suitable to the generation of factor overlays, and if computer assisted methods are to be used successfully, their results must be processed in a manner which changes them into a form which is more compatable with factor overlays. This requires spatial low pass filtering, and occasionally severe filtering. Factor overlays are very simple in spatial structure and contain only enough information to locate and describe a feature of tactical significance. Since overlays are to ultimately be combined, their simplicity becomes even more important. The combination of spatially complex overlays could produce a result containing too much information to be effectively understood and utilized. Production of overlays from computer generated thematic maps generally requires simulating the integration and feature selective filtering performed subconsiously by an image interpreter. Specifically it required the straightening of object boundaries, mapping the object in a contiguous manner, and the removal of small or unimportant features.

In Phase 1 of this project, efforts were restricted to the application of currently available software tools, eliminating the possibility of a truly effective theme filter. The need for one was, however, clear in the earliest stages of the project. Being able to "see the forest through the trees" was not a cliche, but was the definition of a major problem. Extracting forest boundaries meant elimination of high resolution microstructure within the desired feature without objectionably modifying the boundary.

The approach used in Phase 1, was basically to apply an image smoothing filter to the image as a preprocessing function. The smoothed image was then used to generate binary masks of various features after the removal of feature microstructure. Then the element extraction processes, which generally required use of microstructure and high resolution was performed on the original data. Themes generated were then restricted

to an area of the mask to eliminate misclassification errors which would otherwise be widespread and unacceptable. While the overall effort was successful, there were serious problems that were left to be solved. The most serious was that smoothing transformed all discontinuities or feature edges into smoothly flowing ramps. Generation of several of the required masks resulted in a situation where adjacent thematic features did not compliment or butt each other properly, with the area near the feature butt line, corresponding to the high slope region of the ramp, usually not being placed into the proper mask. A second problem was that the smoothing treated all objects equally as opposed to considering the spatial characteristics of individual classes. For example, narrow features such as roads, were obliterated by the heavy smoothing needed to map the forest. Lengthy work arounds were used where several degrees of smoothing were applied, but this was not an effective solution.

As a result of the Phase 1 effort, a series of requirements were identified for a post processing filter to work not on grey level images, but on the themes generated from the images. The advantages of the post processing are many and are basically related to the fact that all resolution is maintained and utilized as needed during the element extraction process. Only after the image has been reduced to the desired themes, is the filtering applied. Post processing solves the mentioned ramp problem by totally avoiding the generation of ramps, and the problem of selective filtering is handled more easily after categorization as the features have been identified.

Processing problems to be solved by a post processing filter:

- <u>Theme Simplification</u> - The severity of the filtering action should be user specifiable to achieve the desired amount of boundary smoothing. Capabilities should include extreme low pass filtering.

- <u>Theme Selective Filtering</u> - All themes are not created equal, and the filter must treat individual themes in individual manners. Methods must be provided for the restriction of growth, shrinkage or change for any particular theme.

- Theme Butting - The filter should be able to annihilate areas between adjacent features of interest which commonly result from classification errors or from the presence of incidental features such as shadows.

- Contiguous Area Thresholding - The filter should be capable filtering by object size to allow removal of islands and/or holes from any designated theme.

In the interest of reasonable execution times, the use of box filtering techniques were nearly dictated for the theme filter. The filter must basically count the number of times each theme occurred in each local neighborhood. Each of the themes is then ranked, based on its local occurrance rate in conjunction with operator designated replacement constraints, as a candidate for being replaced and for being a possible replacement for others. The key theme (the on ioned in the center of the filter area at any given time) is then replaced by the theme at the top of the ranking hierarchy. In the absence of operator input constraints the key theme is replaced by the one most common in the neighborhood. To perform this operation means tabulating running areal totals for each of the themes being processed, and could amount to an excessive number of calculations with clumsy implementation. Box filtering, with its ability to operate with filters of any reasonable size without appreciable execution time penalties, is ideally suited to the task. Keeping running totals with box filters even required less computation than their usual use, which is keeping running averages. As with the running averages, the minor artifacting introduced by the use of a square spatial operator, is outweighed by the computational efficiencies.

Figure 13 shows the theme filter main program, and Figure 14 shows the data structure used. Note that the program is set up to handle eight themes, but nine different categories. Theme #0 refers to uncategorized pixels or those not in any of the eight themes. In retrospect, even another category marking multiply classified pixels or those contained in more than one theme would have been useful.

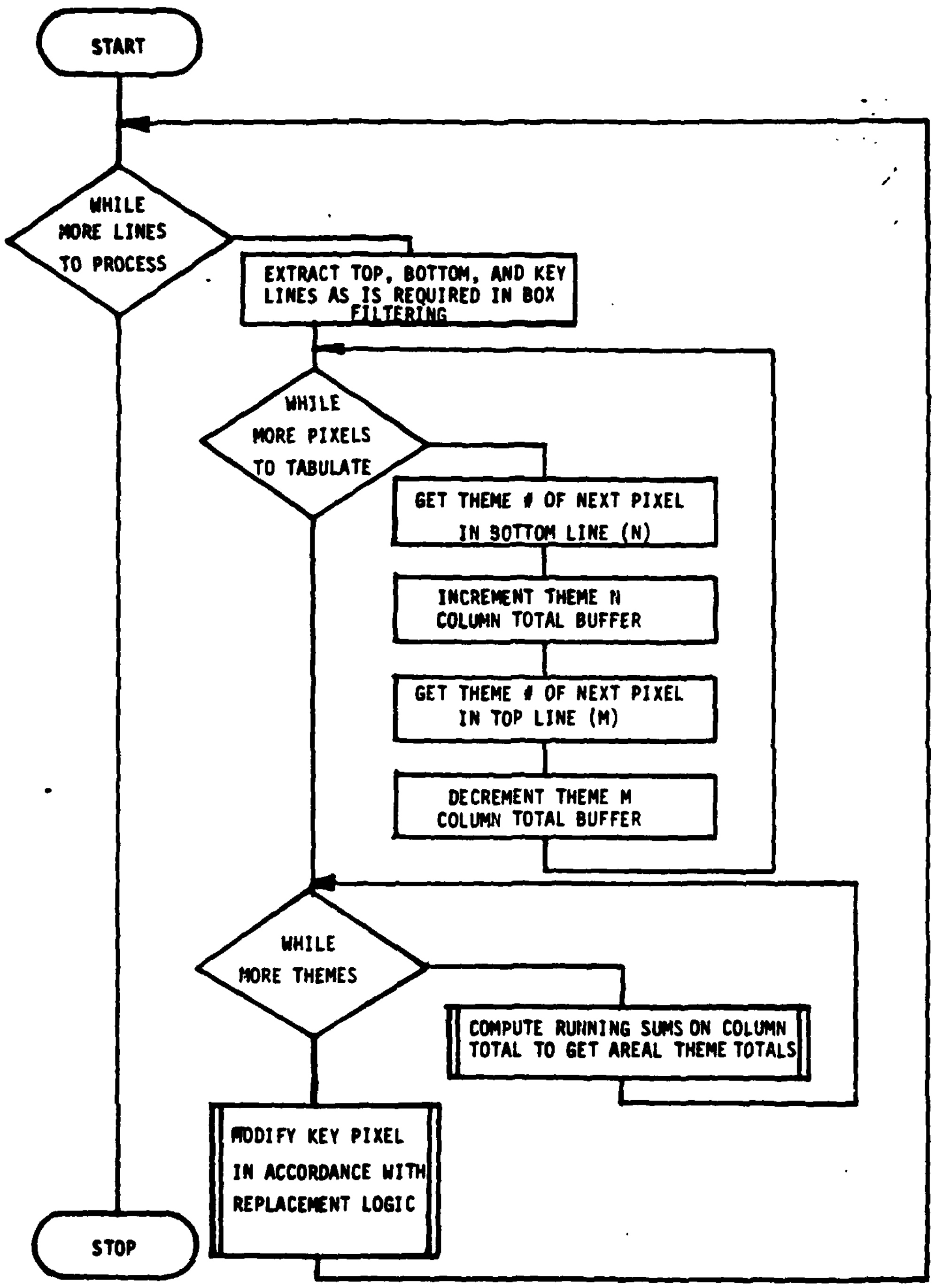


Figure 12. Theme Filter

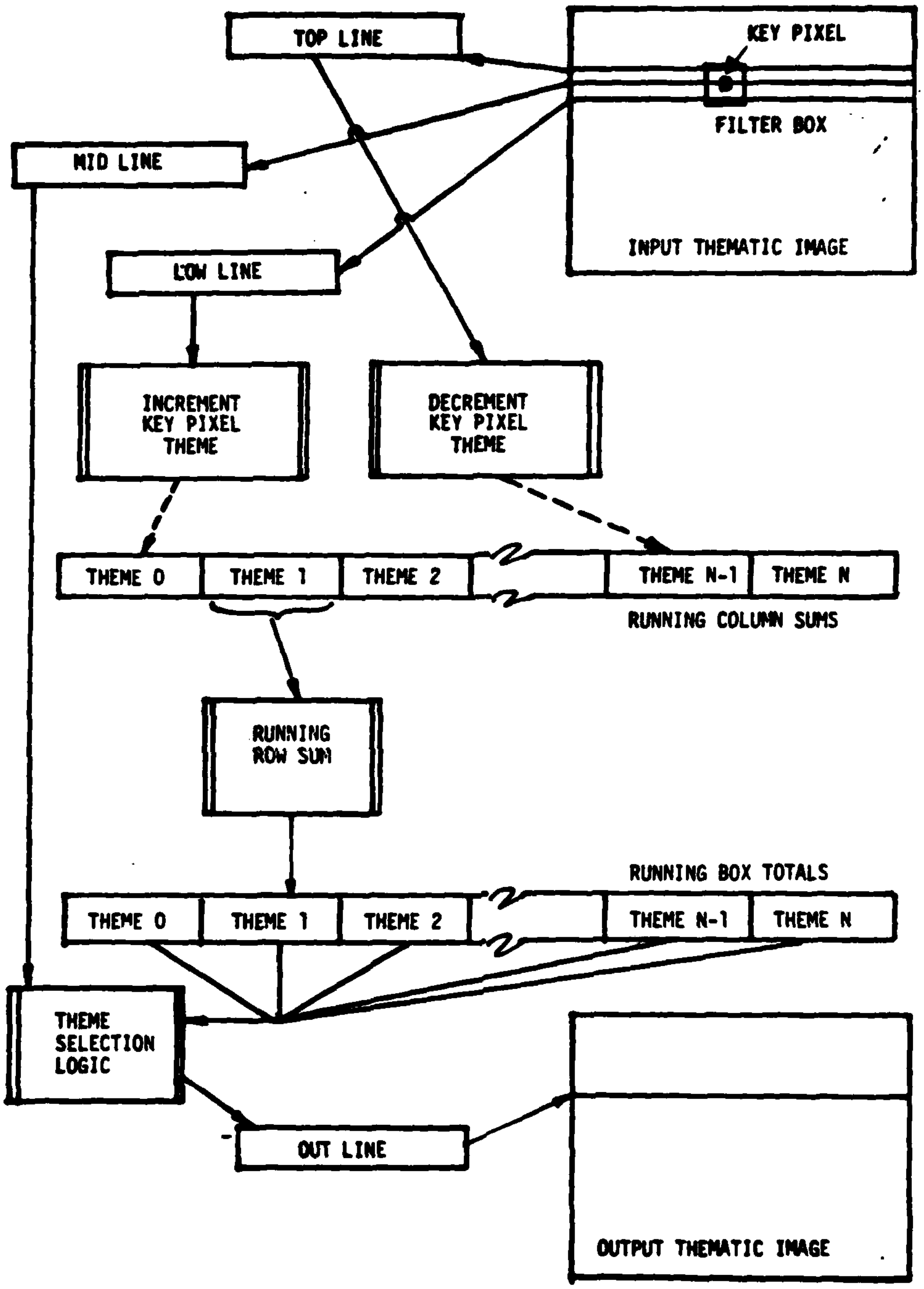


Figure 14. Theme Filter Data Structure

The box filtering approach is not unique in theme processing, but as commonly used it is not sufficient to satisfy the theme processor requirements. Used without replacement constraints, it is not versatile enough to perform all the desired theme manipulation possibilities. Box filtering is simply an effective method of providing information to the real heart of the theme processor, the theme replacement logic.

Interactive control of the theme filter is accomplished by setting theme population thresholds which must be exceeded before a given theme is modified, with separate thresholds for theme growth and for theme shrinkage. The filter can thus be set up to treat themes uniquely when needed. Whenever the shrink threshold is passed, the key pixel is replaced by the most common growable theme. See Figure 15.

Considerable effort went into the operator interface. Its purpose is to assist the operator in the task of setting up fairly complex filtering options. The filter has numerous input parameters, whose input is tedious and has a high potential for error. Also, since the filtering action on a given theme is determined not only by the thresholds directly associated with that theme, but also by the thresholds of its possible replacements, the interaction between these input parameters is not always obvious. The interface is designed to minimize confusion along with tedious repetitive input. The filter is set up to operate in an iterative manner. Allowing either continuing with the filtering or reworking the last operation if undesirable results appear. The filter executes fast enough so that an operator can observe and analyze the result, and then rerun the filter with modified thresholds or box size to better achieve his purpose.

Figure 16 is a verification display for the interface, listing current threshold settings along with a computer generated verbal estimate of the resulting filtering action.

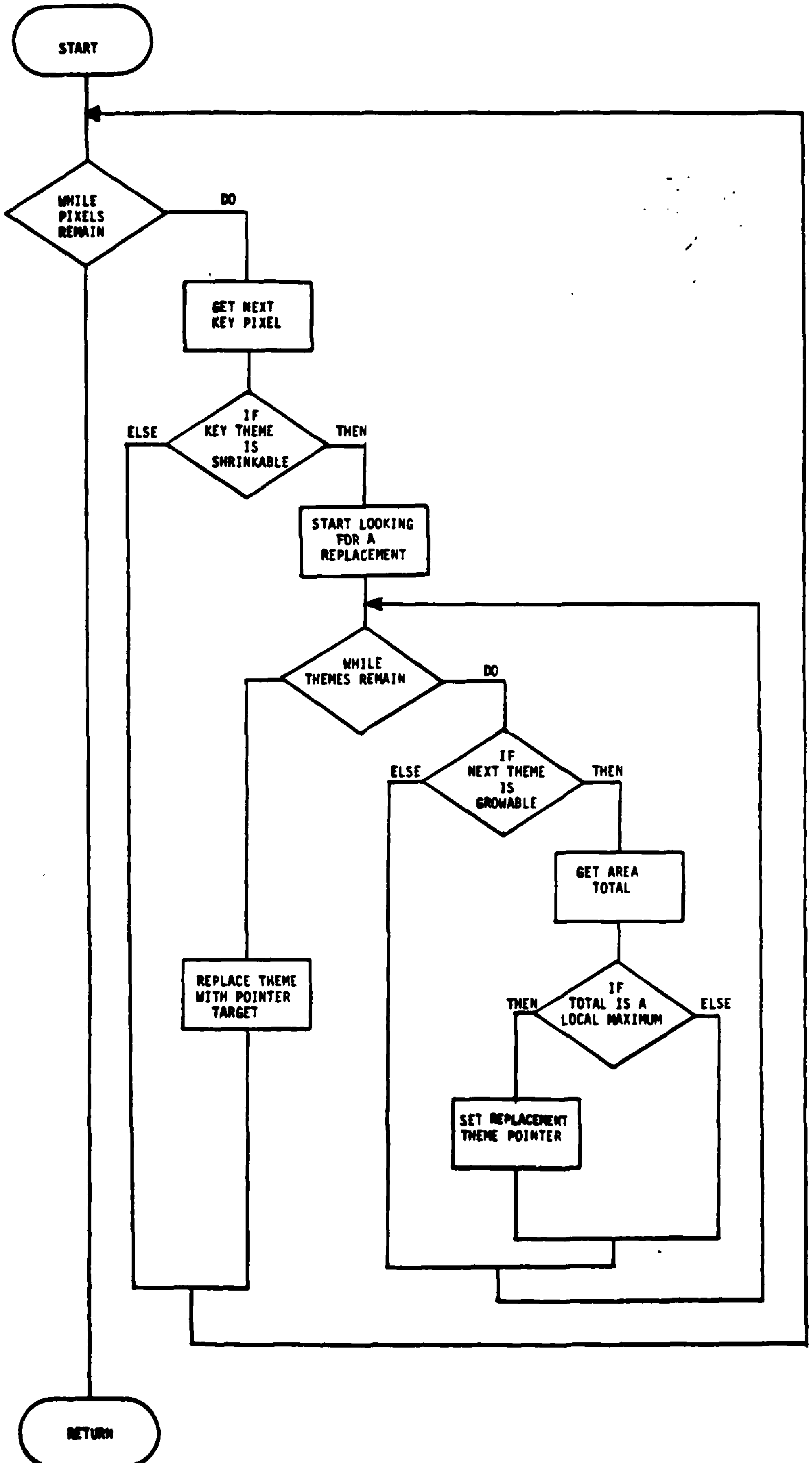


Figure 15. Theme Filter Replacement Logic

THE FILTER OPERATES BY TABULATING THE NUMBER OF TIMES EACH INDIVIDUAL THEME OCCURRS INSIDE A MOVING RECTANGULAR AREA. FOR EACH FILTER POSITION, IF THE THEME IN THE CENTER OF THE AREA IS ALLOWED TO SHRINK, IT WILL BE REPLACED BY THE MOST COMMONLY OCCURRING AMONG THOSE ALLOWED TO GROW. WHENEVER A SUITABLE REPLACEMENT IS NOT FOUND, THE THEME IS LEFT UNALTERED.

SELECTIVE THEME REPLACEMENT IS CONTROLLED BY ENTERING THRESHOLDS EXPRESSED AS PERCENTILES OF THE FILTER AREA. A SPECIFIC THEME IS ALLOWED TO SHRINK IF THE NUMBER OF TIMES IT OCCURRS IS LESS THAN IT'S SHRINK THRESHOLD, AND IS ALLOWED TO GROW IF OCCURANCES EXCEED IT'S GROW THRESHOLD. A NEGATIVE THRESHOLD VALUE CAUSES A THEME TO BE TOTALLY REPLACED.

11 X 11 FILTER 09-FEB-82 10:03:27

THEME THRESHOLD PERCENTILES

THEME NUMBER	SHRINK IF BELOW, %	GROW IF ABOVE, %	FILTERING ACTION	
0	***	***	REPLACE	______
1	100	0	SMOOTH	______
2	100	0	SMOOTH	______
3	0	100	NO CHANGE	______
4	100	100	SHRINK ONLY	______
5	0	0	GROW ONLY	______
6	15	100	REMOVE ISLES	______
7	0	85	REMOVE HOLES	______
.8	5	95	CLEAN NOISE	______

CHANGE THEME THRESHOLDS? (Y)ES, OR (N)O >

Figure 16. Theme Filter Operator Interface

D. Use of the Theme Filter

In the pursuit of water resources element extraction it was established that the theme filter was effective in solving a host of image categorization problems. The most severly limiting factor in its usage seems to be the limited ability of the analyst to correctly forcast the action of the filter and to be able to specify the set of interrelated thresholds needed to perform the desired function. Several common image categorization problems, along with suggested and proven solutions making use of the theme filter, are listed below:

1. Capabilities

a. General Theme Smoothing

Application: Spatially simplify theme so that they may be depicted by line maps.

Procedure: Run filter with default thresholds so that each theme is replaced by the locally most common. For best results, run the filter twice to insure no linear artifacts are generated by the rectangular filter.

b. Annihilation of Certain Themes

Application: Often it is desirable to totally eliminate irrelavent themes (shadows for example).

Procedure:

- Enter a negative threshold for the theme to be annihilated.
- Leave all other thresholds default

The theme filter has a replacement flag set by entering a negative threshold. This flag stops the theme from being considered as a possible output.

c. Preserving Certain Themes While Filtering Others

Application: Very accurately mapped themes, or sparce filament themes (roads, etc.) should not be changed.

Procedure:

- Set grow above threshold to 100% so that theme will not grow.
- Set shrink below threshold to 0% so that theme will not shrink.

d. General Noise Cleaner

Application: Most spatially derived themes are improved by eliminating isolated pixels which often represent noise.

Procedure: Set thresholds for all themes to grow above 33% and shrink below 11%, then run a 3 x 3 pixel filter.

The 11% threshold locates totally isolated pixels, while the 33% threshold insures there is a locally dominant class before replacement. By enlarging the filter size this operation can be used to remove larger areas (islands and holes).

e. Resolving Theme Overlap or Underlap

Application: Image classification virtually always results in some error as signatures are so unique. Problem areas often show up as unclassified or multiply classified pixels. A satisfactory solution is to remap these poorly classified pixels on a purely spatial basis.

Procedure:
- Place pixels to be resolved into a separate theme
- Set replacement flag for that theme
- Leave default thresholds for all other themes

2. Processing Example

To demonstrate the filtering capability, one of the panchromatic scenes used in the first phase of this project was recategorized with considerable improvement. The following processing sequence was followed and results are shown in Figure 17.

a. The scene was categorized on the basis of gray level and microtexture. This task was some what easier than with the previous effort because the filter will contiguously map locally predominant classes allowing less descriminative signatures to be used.

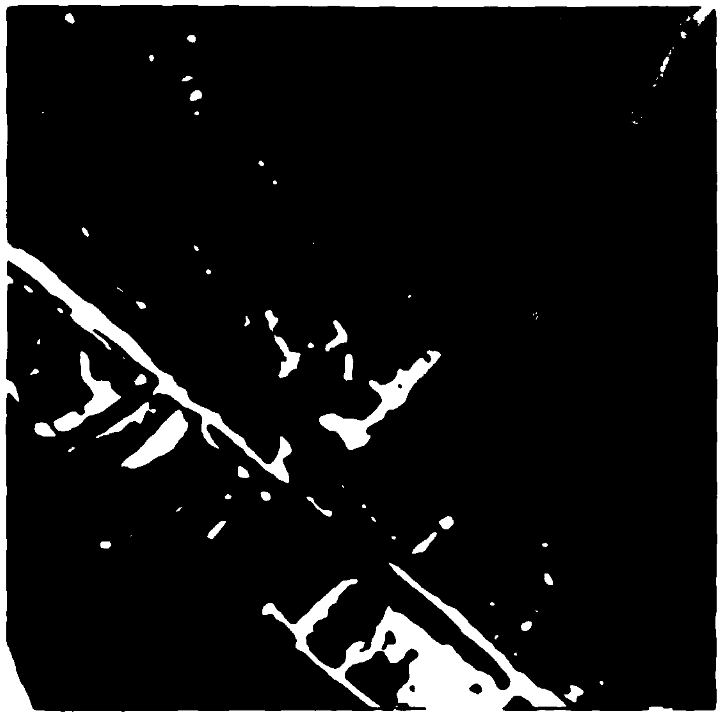


Figure
Theme From Phase 1

Figure
Manual Interpretation
From Phase 1

Figure 17. Comparison of Fort Belvoir/Woodbridge Vegetation Themes Derived by Interactive Computer and Manual Interpretation.

b. Areas categorized as shadows were eliminated by the filter based on operator interpretation. Two passes were used: (1) when a shadow occurred within a short distance to water, the shadow was changed to water producing considerable improvement in the river shoreline and bridge area, and (2) shadows not near water were mapped to the locally most common class.

c. The scene was then smoothed by a 5 x 5 pixel filter to produce a categorization comparable to what must be done by a photo interpreter. The "bare" category was restricted from change so that the roads would be preserved. See Figure 18A and 19.

d. To demonstrate severe filtering, the categories were then processed with an 11 x 11 pixel filter. See Figure 18B and 20. allows virtually any degree of filtering to be achieved with about the same execution time. In working situation, the desired amount of filtering might be determined by scene complexity, desired scale, or intended use.

e. Both of the products were reduced to line maps via Golay processing (essentially automated). Each theme was outlined, the outlines were combined to produce double width lines, and then the lines were shrunk to a single pixel width.

Figure 18A

Theme Filter Results

Figure 18B

Simplified Themes

Figure 18. Results of Theme Filtering

Figure 19. Outlines of Vegetation and Water Themes

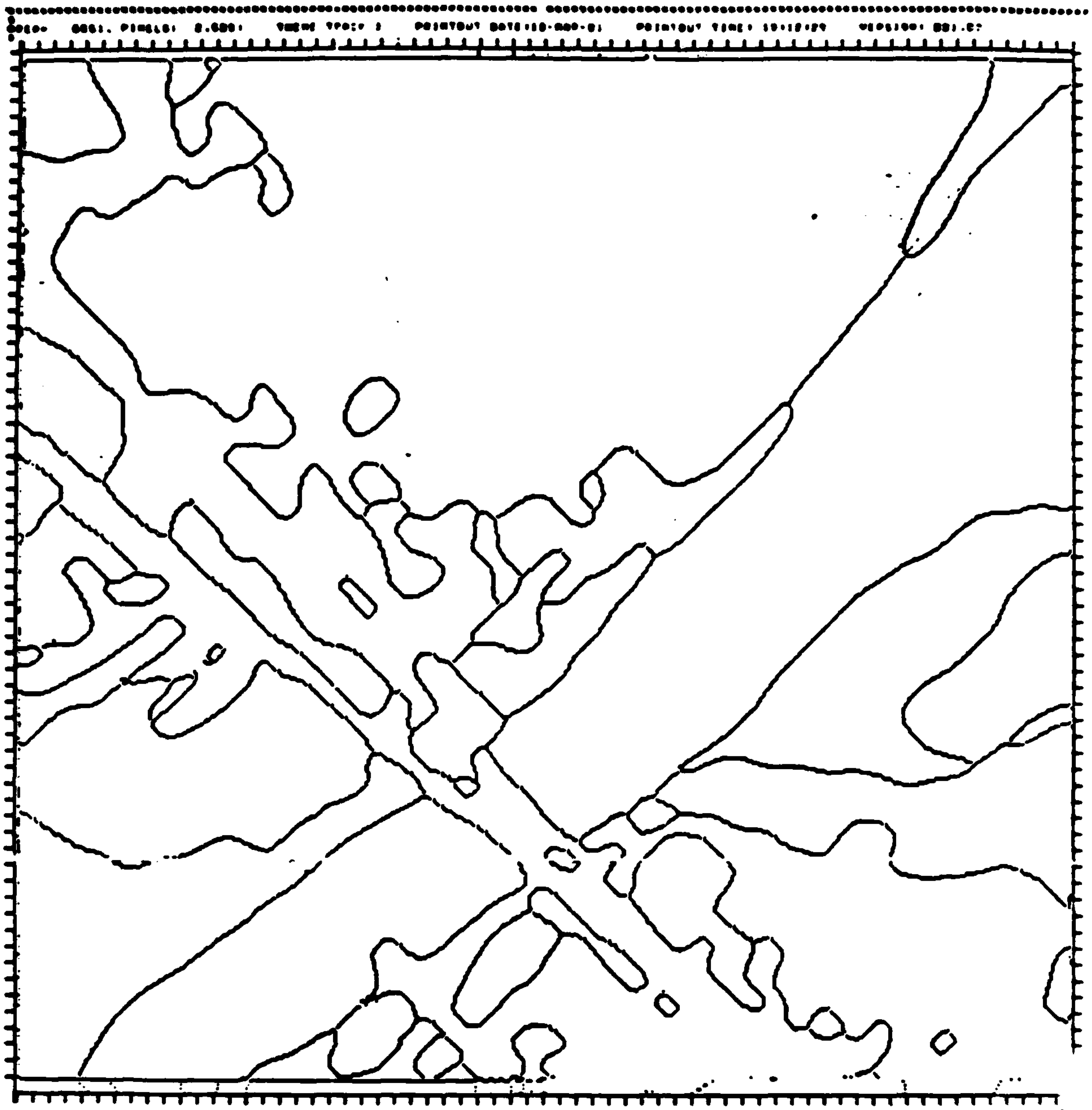


Figure 20. Outlines of Simplified Themes

IV. RESULTS

A. Extraction of Watercourses and Associated Elements

1. Watercourses

a. Extracting Watercourses From Panchromatic and Thermal Imagery

The Knobbs Creek imagery was selected for analysis and development of techniques because it contained data coverage from both thermal IR and panchromatic photographic sensors. Knobbs Creek is a good test for watercourse classification in that the coverage contains a wide range of watercourse widths. It provides a difficult test as the creek is almost 100% surrounded by large trees. This does not provide, however, the best diversity of surrounds with which to test watercourse identification. The ideal test would involve a scene having many watercourses surrounded by a variety of land cover, and having a wide range of watercourse widths.

Knobbs Creek, being surrounded by forest, creates problems for both machine and manual analysis. The creek is obscured by forest when watercourse width is less than approximately 5 meters. In these situations, analyst inferred watercourses will be necessary. Further, the determination of dry gap widths is hindered by the encroaching forest and by shadows from adjacent forest which extend across the watercourse. The shadows create an additional problem for machine analysis by completely or partially covering the watercourse (depending on watercourse width or direction). This requires the machine analyst to derive a signature for water and a signature for shadow which must be added together to enable identification of the entire watercourse. Unfortunately, the signature contains areas which are not watercourses (forest shadows within the forest).

Eight (512 x 512 pixel) subscenes were digitzed at GE DIAL from three of the 1:100,000 scale thermal IR transparencies of Knobbs Creek. Four of these were digitized from 4 in. x 4 in. areas in the transparency (ground resolution approximately 2 meter) and four from 1 in. x 1 in. areas (ground resolution approximately 0.5 meter). In addition, four (512 x 512 pixel) subscenes were digitized at GE DIAL from the 1:250,000 scale panchromatic

photographic image of Knobbs Creek. Two of these were digitized from 4 in. x 4 in. areas (ground resolution approximately 4 meter)* and two from 1 in. x 1 in. areas (ground resolution approximately 1 meter)*. These resolutions were chosen for initial tests of watercourse identification to provide an appropriate range of resolution for testing the spatial filters used in watercourse extraction.

The initial step in the investigation was to identify whether gray level density slicing would be sufficient to identify watercourses. Several problems were encountered in this step:

- This thermal IR imagery appears to contain dark and light bands which are apparently caused by automatic gain and offset in the sensor which are adjusted by scan content. This has the undesirable effect of causing forest and watercourse signatures from different areas of the scene to appear similar and thus be recognized simultaneously.
- Both thermal IR and panchromatic photographic digital images exhibit a shading across the image which cause signatures to overlap.

Gray level density slicing of either the thermal or the panchromatic photographic digital images, in order to extract watercourses, produced unacceptable results. Figures 21 and 23 show the watercourse themes extracted from the panchromatic photographic images, and similar poor results were obtained with the thermal images.

At this point, several spatial filters were applied to all digital images. The images resulting from use of these filters have high spatial frequency noise, however, which requires a subsequent smoothing operation. Best smoothing results appear to be produced with the 7 x 7 pixel moving average. Some of the spatial operations tested include: 5-pixel cross min-max texture; the original image minus a 7 x 7 pixel local average; 3 x 3 pixel Laplacian.

---

* In this report, the term resolution is used to describe the actual dimensions of the smallest resolvable objects in an image.

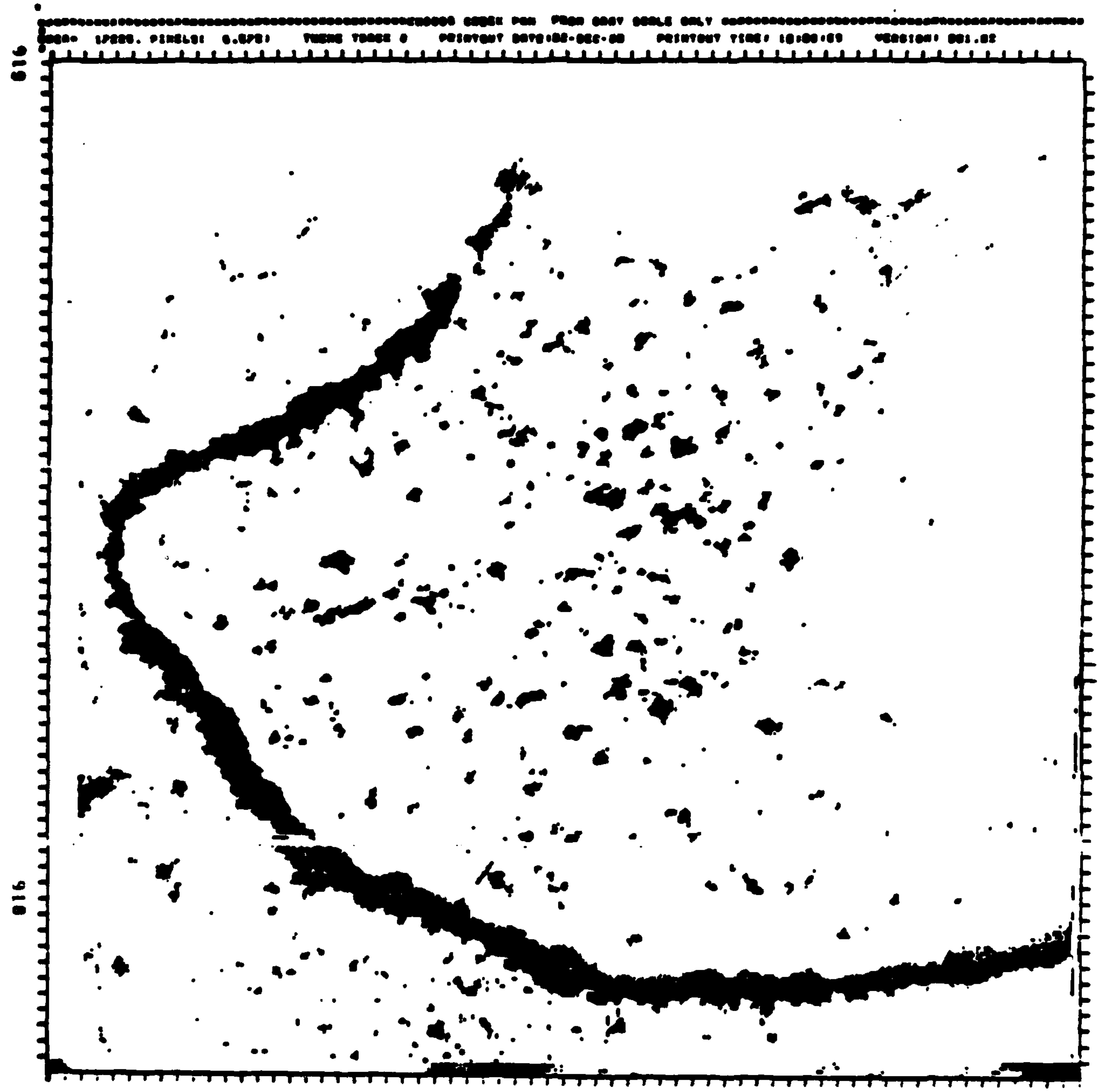


Figure 21. Watercourse Extraction via Level Slicing The 1-meter Resolution Panchromatic Photographic Image.

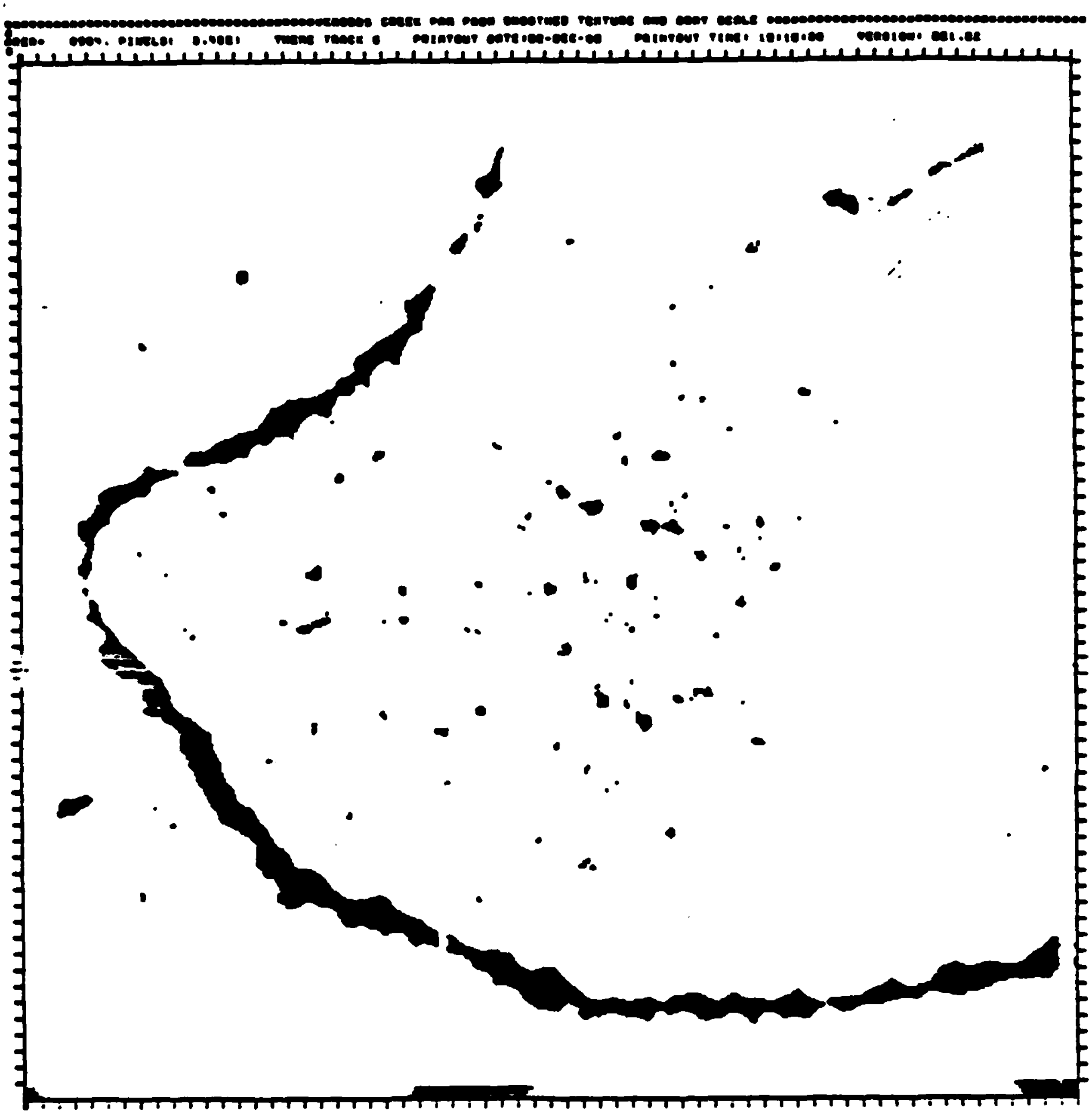


Figure 22. Watercourse Identification via Joint Use of the 1-meter resolution Panchromatic Photographic Image and the Derived Texture Image (Smoothed).

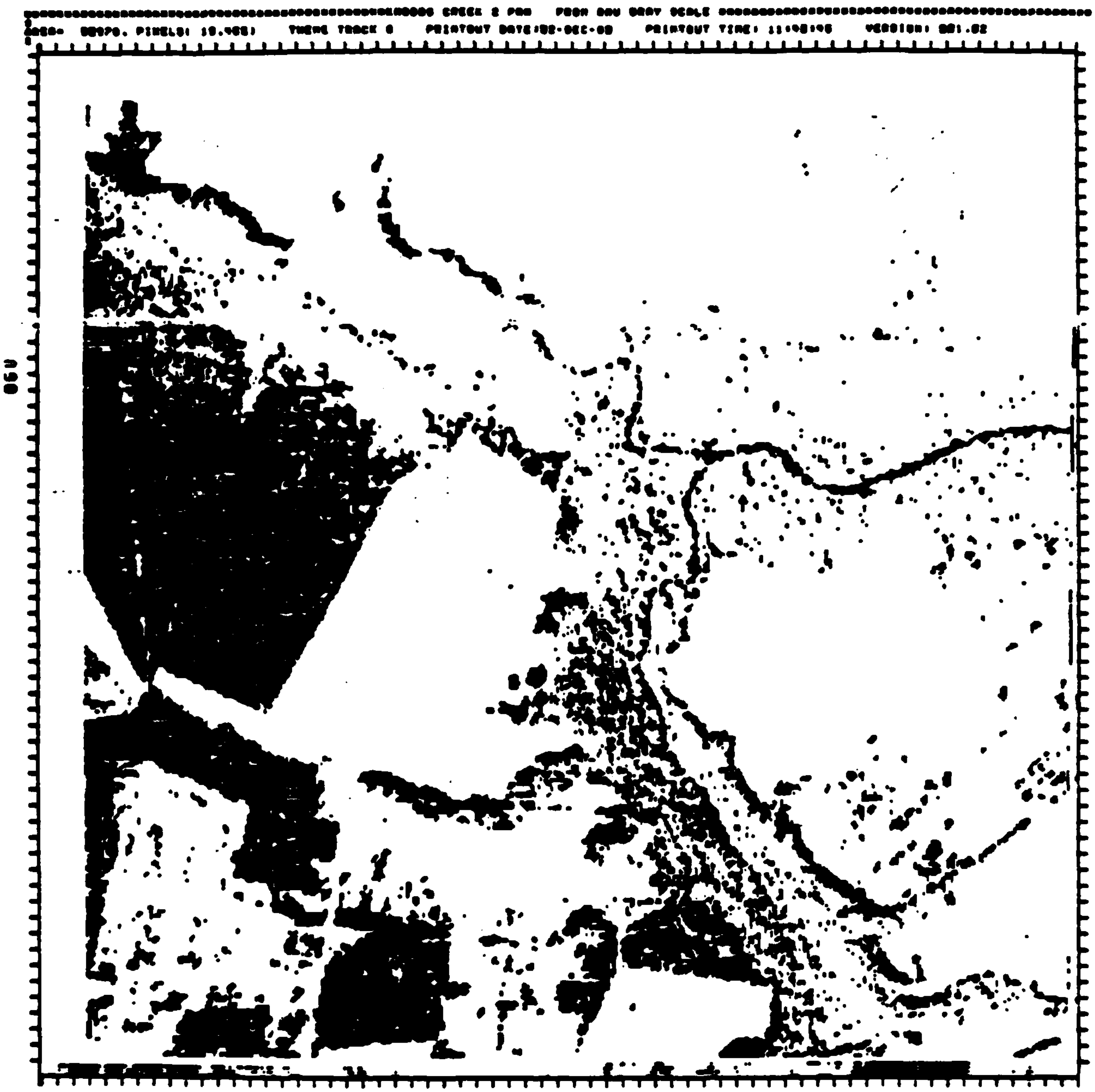


Figure 23. Watercourse Extraction via Level Slicing The 4-Meter Resolution Panchromatic Photographic Image

Results when using the spatially processed images jointly with the original digital images (i.e., two dimensional classification) can be summarized as follows:

- Best results for 1 meter resolution panchromatic photographic and 0.5 meter resolution thermal IR images were obtained with the 5-pixel Cross min-max texture operator.
- Best results for 4 meter resolution panchromatic photographic and 2 meter resolution thermal IR images were obtained with the original image minus the 7 x 7 local average.

Figure 22 shows the result of watercourse identification using the 1 meter resolution panchromatic photographic image and its texture version. In Figure 24, watercourse identification is shown for the case when 4 Meter resolution panchromatic photographic image and the smoothed result of image minus a 7 x 7 pixel local average are used. Figures 23 and 26 show the actual location of the watercourse. Comparison of actual and machine-extracted watercourses indicates shadow areas in both scenes which are falsely included in the watercourse theme.

Similar results were obtained with the digitized thermal imagery.

2. Extracting Watercourses from Radar Imagery

Four subscenes of 512 x 512 pixels depicting X-Band vertical and horizontal polarization; and L-Band vertical and horizontal polarization were made. Figures 25, 26, 27, and 28 are illustrations of the subscenes as displayed on the cathode ray tube (CRT). The data are shown inverted to a negative. Geographic scene locations are shown in Figure 29.

In Figures 30, 31, 32 and 33 theme results were obtained by sensity slicing the data. Figure 34 subscene is the same as Figure 25. It is repeated to allow for comparison relative to the processes subscenes in Figures 35, 36 and 37. In Figure 35, the subscene illustrates the application of the 7 x 7 moving average operator on the X-Band horizontal polarization data. Figure 36 shows the derived associated microtexture image derived from X-Band horizontal polarization data.

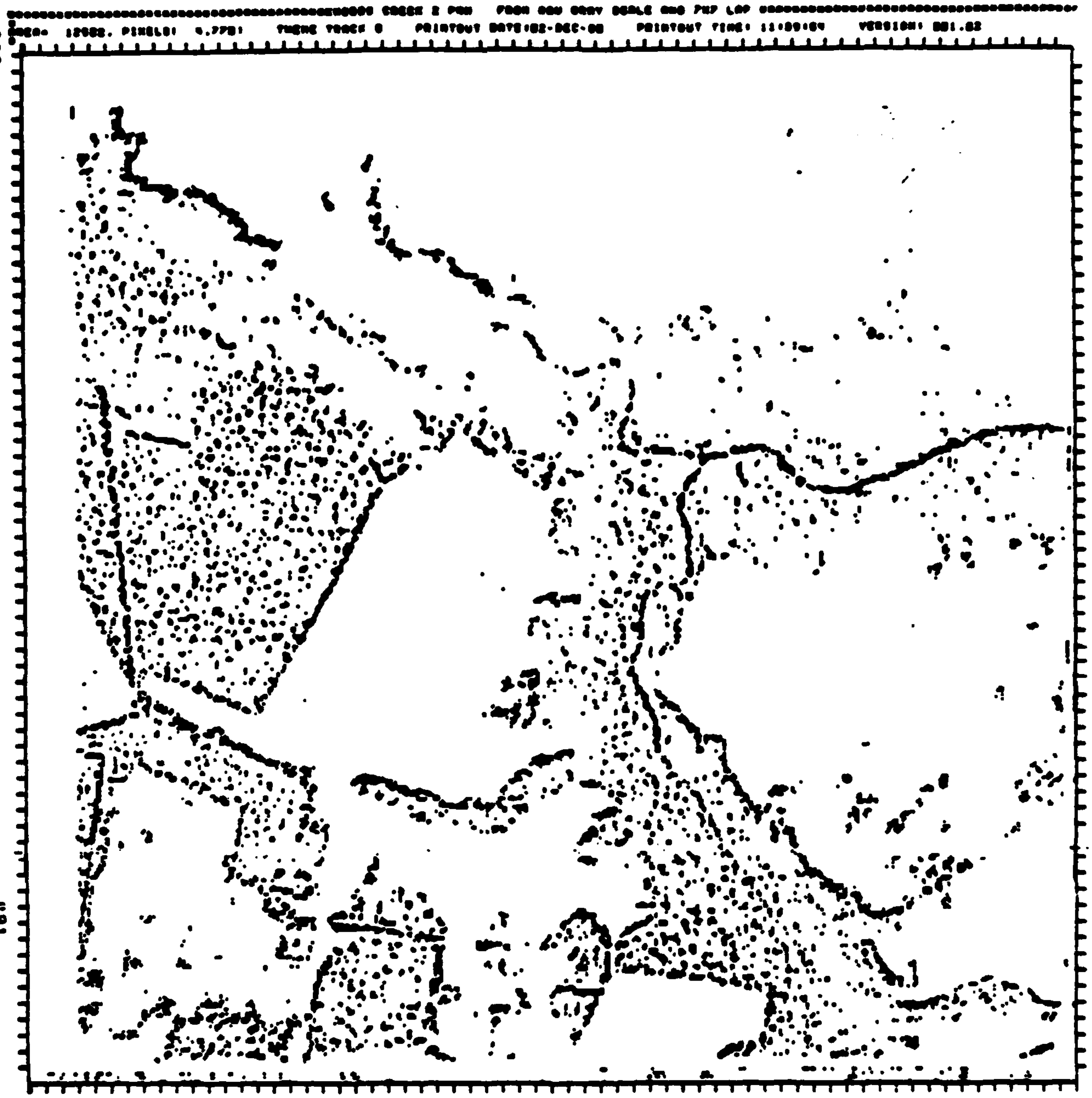


Figure 24. Watercourse Identification via Joint Use of the 4-Meter Resolution Panchromatic Photographic Image and the Result of the Image Minus 7 x 7 Pixel Local Average.

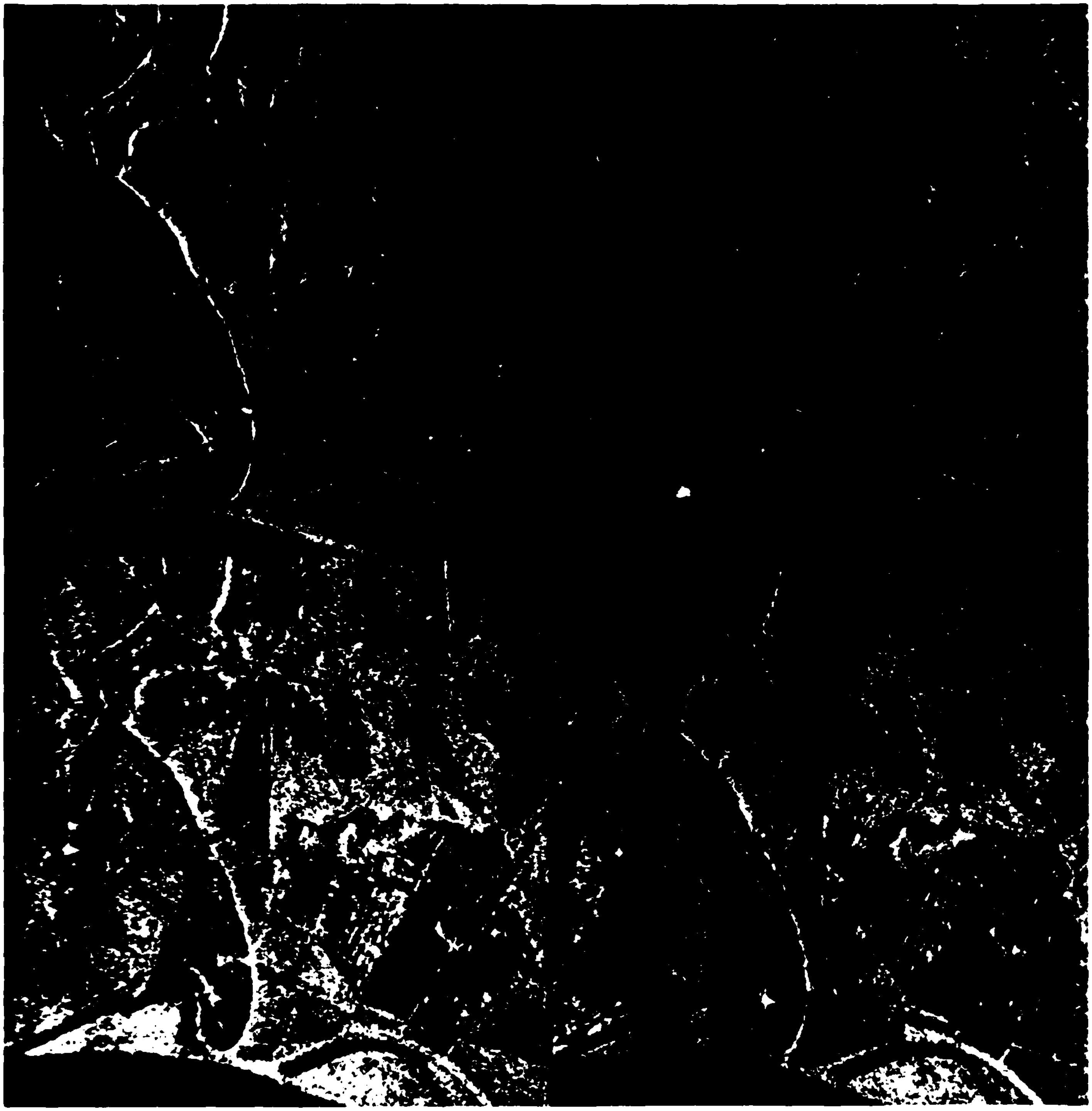


Figure 25 - X-Band Horizontal Polarization Subscene

Figure 26 - X-Band Vertical Polarization Subscene

Figure 27 - L-Band Horizontal Polarization Subscene

Figure 28 - L-Band Vertical Polarization Subscene

CHARLESTON, WEST VIRGINIA
EASTERN UNITED STATES 1:250,000

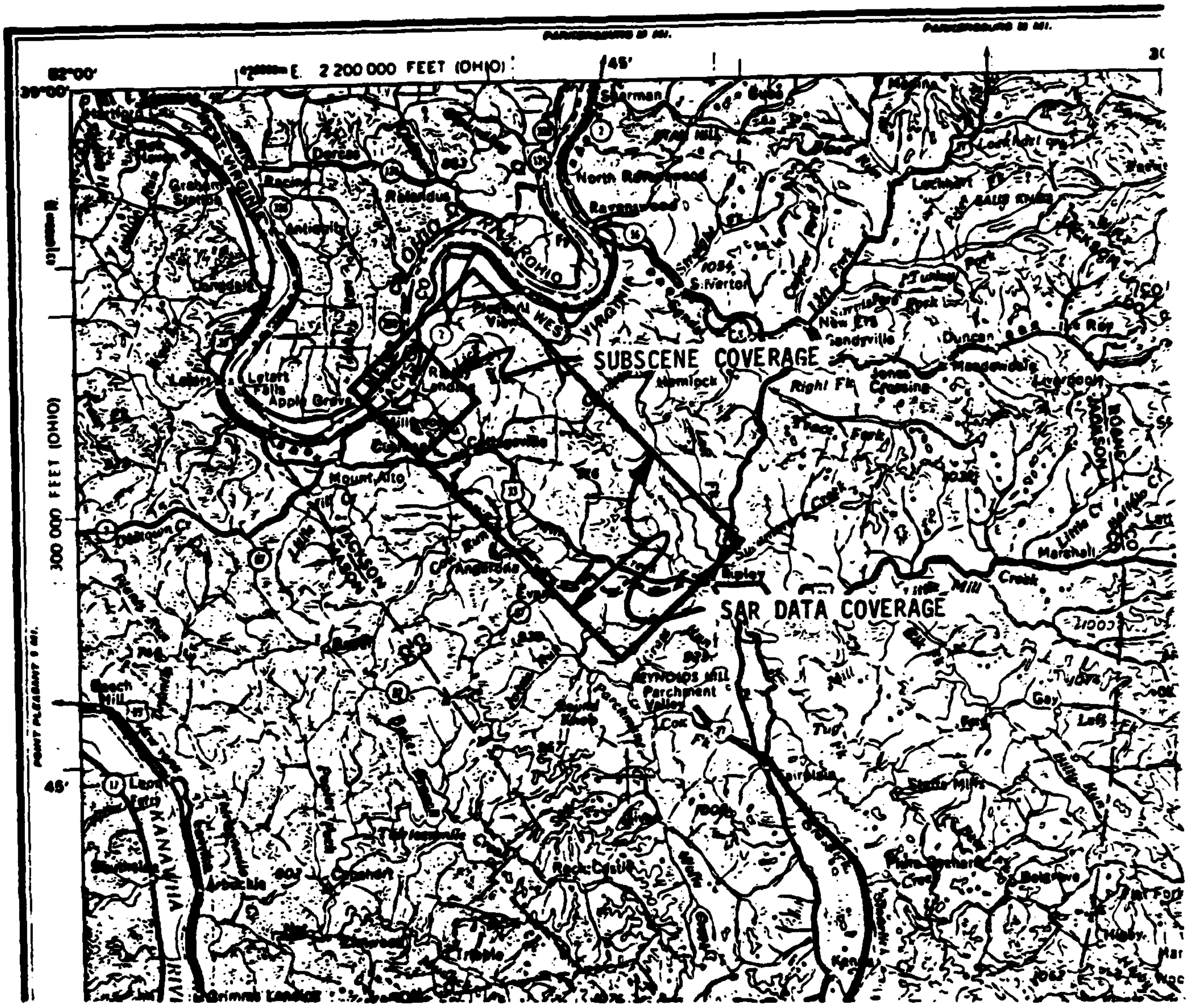


Figure 29. SAR Data and Subscene Locations.

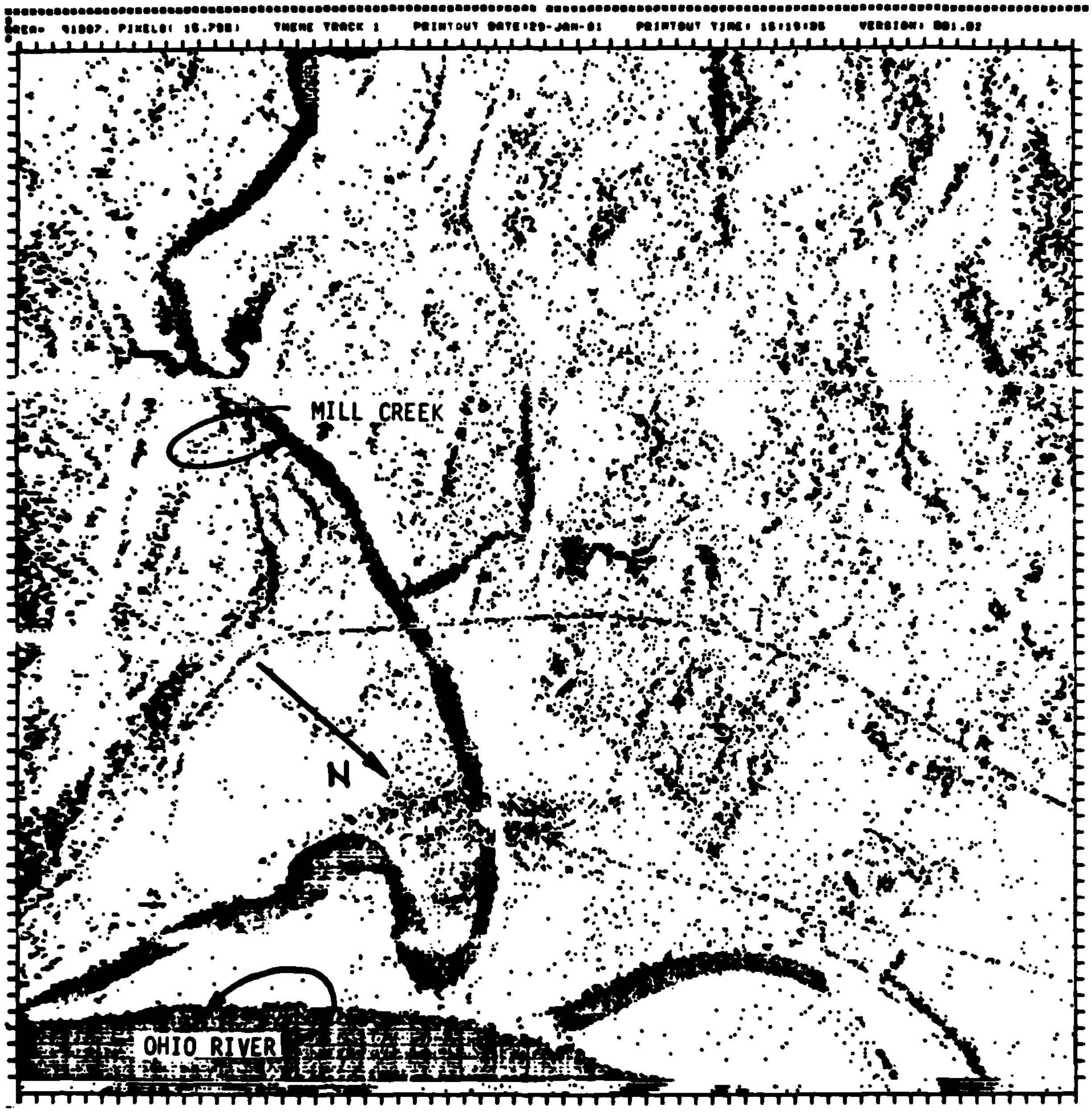


Figure 30. Water Theme Sliced From X-Band SAR w/Horizontal Polarization.

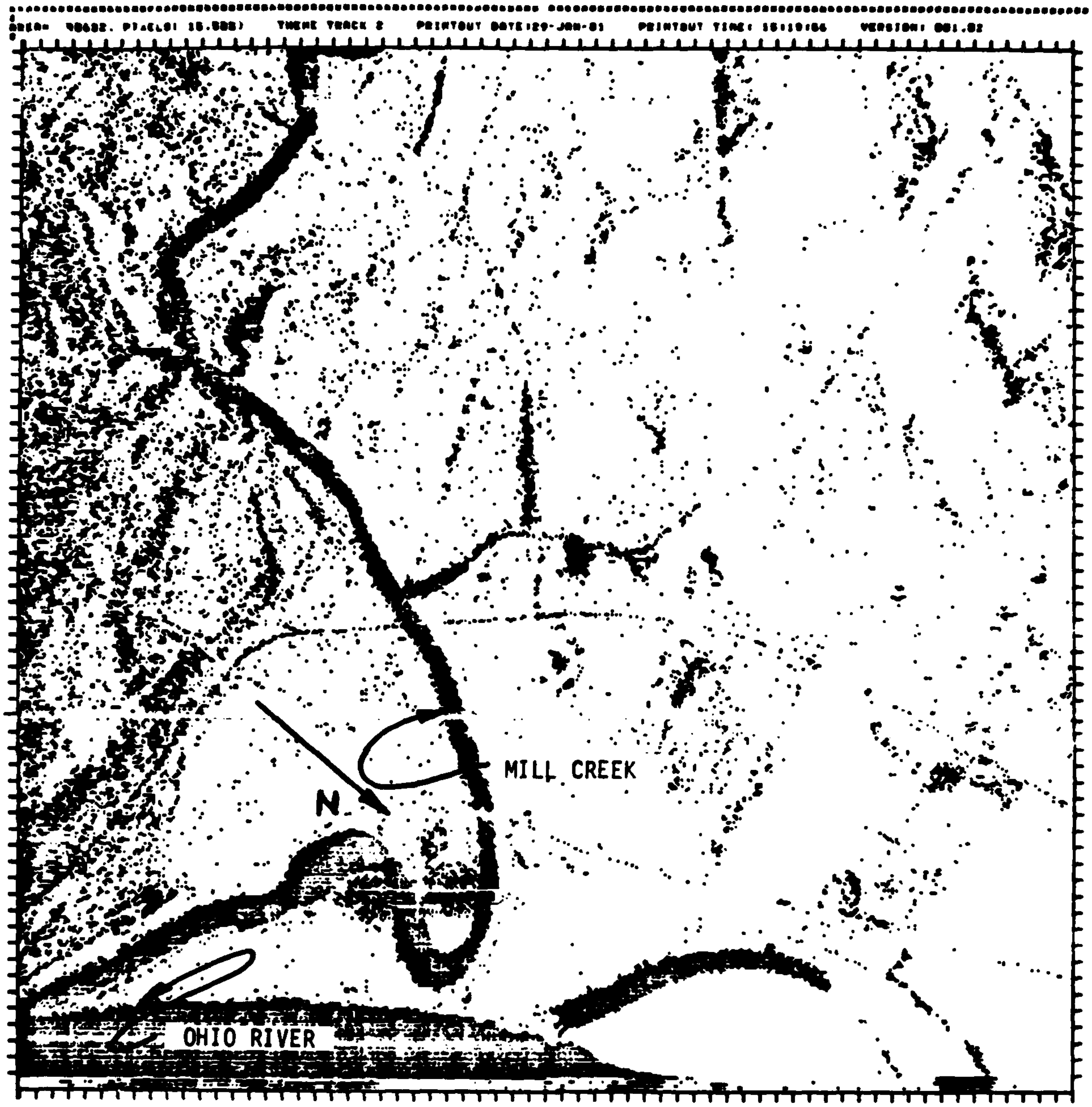


Figure 31. Water Theme Derived From X-Band SAR w/Horizontal Polarization.

Figure 32. Water Theme Derived From X-Band SAR w/Horizontal Polarization.

Figure 33. Water Theme derived from L-Band SAR w/vertical polarization.

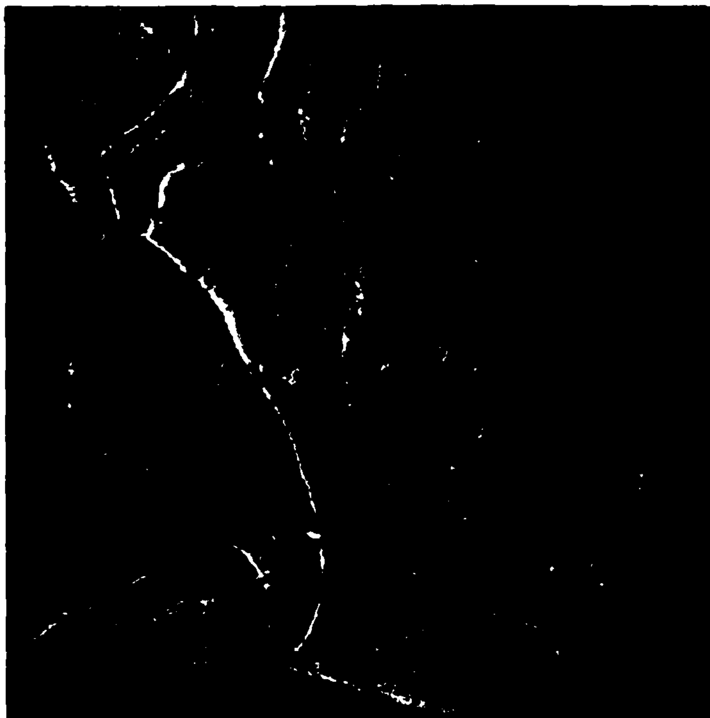


Figure 34 - X-Band Horizontal Polarization Subscene (Same as Figure 1)

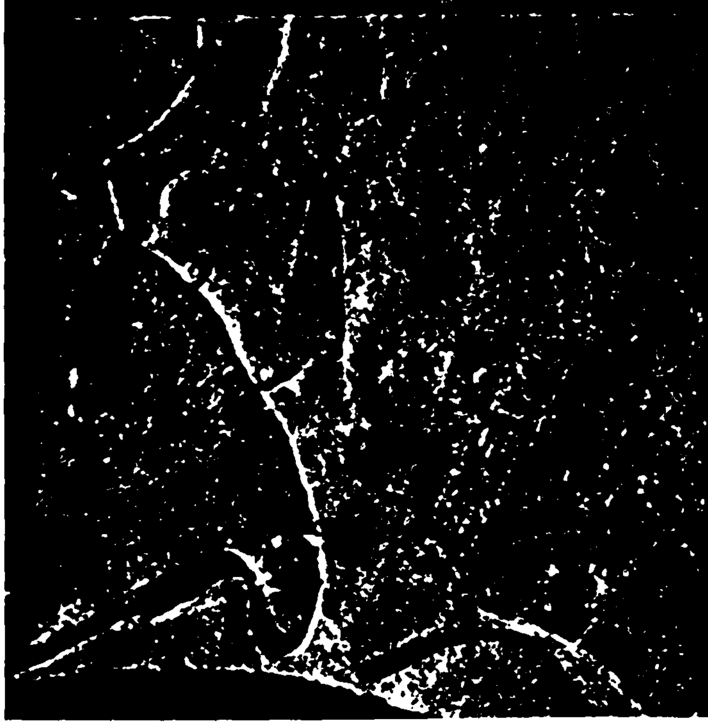


Figure 35 - X-Band Horizontal Polarization Data Smoothed by a 7x7 Pixel Moving Average Operator

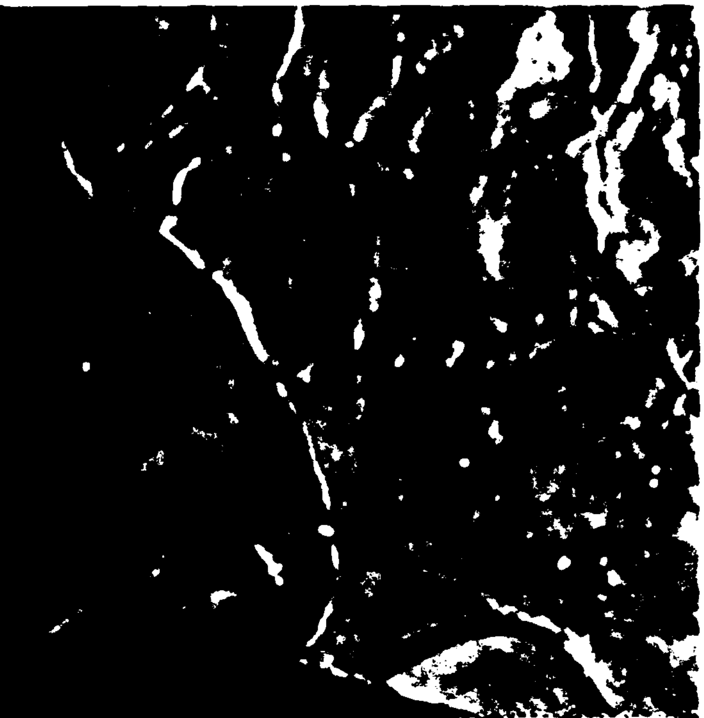


Figure 36 - Associated Microtexture Image

Figure 37 - Microtexture Image Data Smoothed by a 7x7 Pixel Moving Average Operator

Figure 37 shows the results of smoothing the microtexture image data with a 7 x 7 pixel moving average operator. In Figure 38, X-Band SAR horizontal polarization data and the associated microtexture image are shown categorized by a two dimensional partitioning classifier. This procedure appears to provide clearer results than density slicing the SAR data only. Figure 39 illustrates the categorization of water courses and water bodies using the two dimensional partitioning classifier with the X-Band SAR horizontal polarization data and the texture data smoothed by a 7 x 7 pixel moving average operator. It appears that another degree of clarification is achieved with this method. In Figure 40, X-Band SAR horizontal polarization data are smoothed by a 7 x 7 pixel moving average operator and density sliced into a theme for water courses and water bodies. Apparently this procedure provides the best definition of watercourses and water bodies.

3. <u>Watercourse Extraction Using Combinations of Dissimilar Images</u>

A data set of panchromatic, low resolution Thermal IR and SAR (two flight directions) images were digitized at 5 meter resolution and registered for Little River and New Begun Creek water courses. Different combinations of data from the set, including microtexture derived data (from panchromatic images) were used for the extraction of water course themes. The low resolution Thermal IR data did not have a significant contribution to the derivation of the watercourse themes. From the analysis of a combination of the panchromatic images and the microtexture (derived from the panchromatic images) data a conflict between shadows and water existed. Radar imagery was used to resolve some of the shadow conflict. Derived watercourse themes for Little River are shown in Figures 41, 42 and 43..

A comparison of watercourse themes which were derived from different combinations of Chapel Creek (upper Little River area) panchromatic, radar, and the microtexture data (derived from the panchromatic images) are shown in Figures 44,45, 46 and 47.

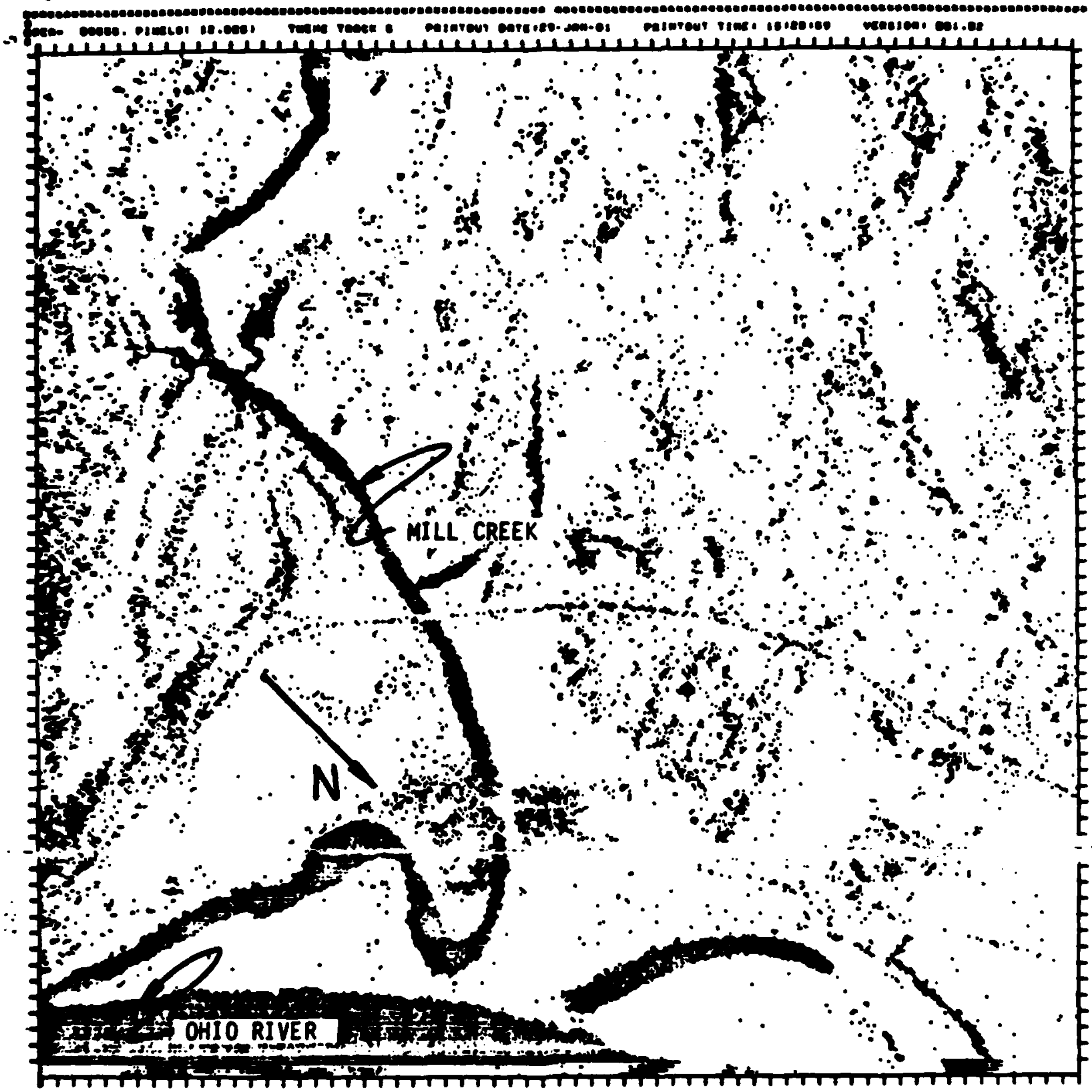


Figure 38. Water Theme Derived from w/Horizontal Partitioning of X-Band SAR w/Horizontal Polarization and Associated Microtexture Image Data.

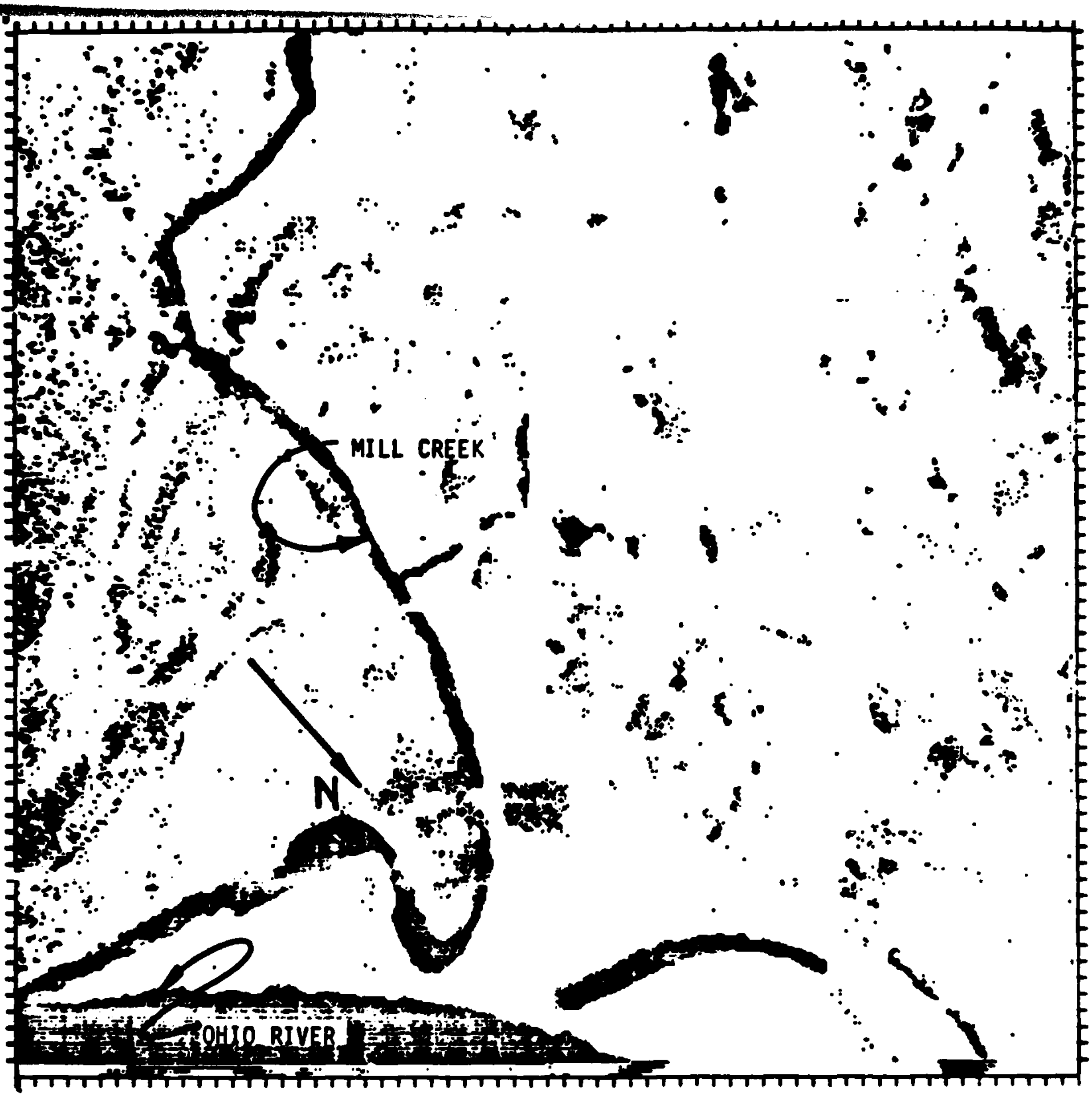


Figure 39. Water Theme Derived From 2-Dimensinal Partitioning of X-Band SAR w/Horizontal Polarization Data and Textured Data Smoothed By a 7 x 7 Pixel Moving Average Operator.

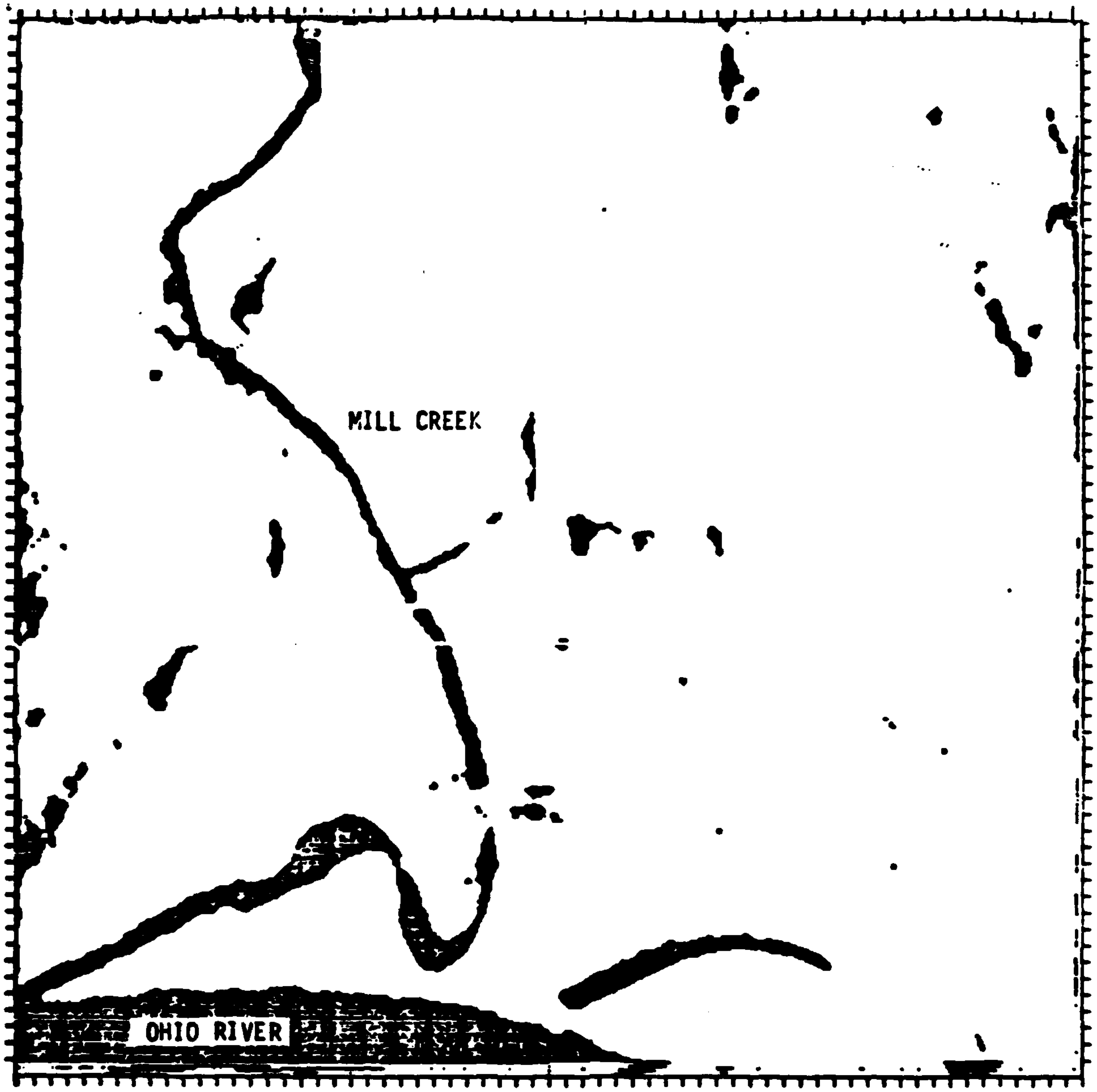


Figure 40. Theme Derived From the 7 x 7 Pixel Smoothed Version of the X-Band SAR Image w/Horizontal Polarization.

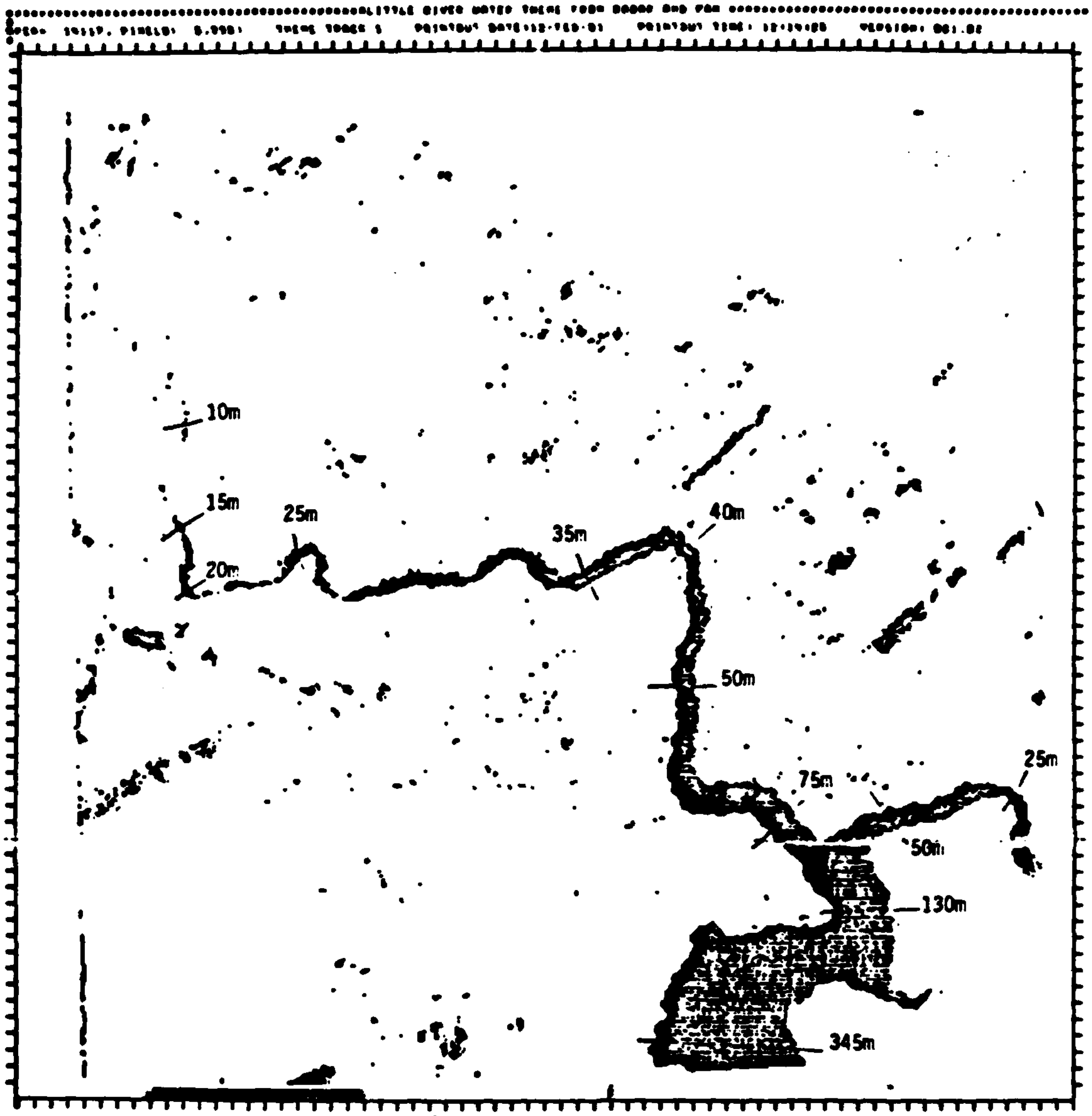


Figure 41. Little River Watercourse Theme Derived from Panchromatic Photography and SAR Data and from the Microtexture Version of the Photography.

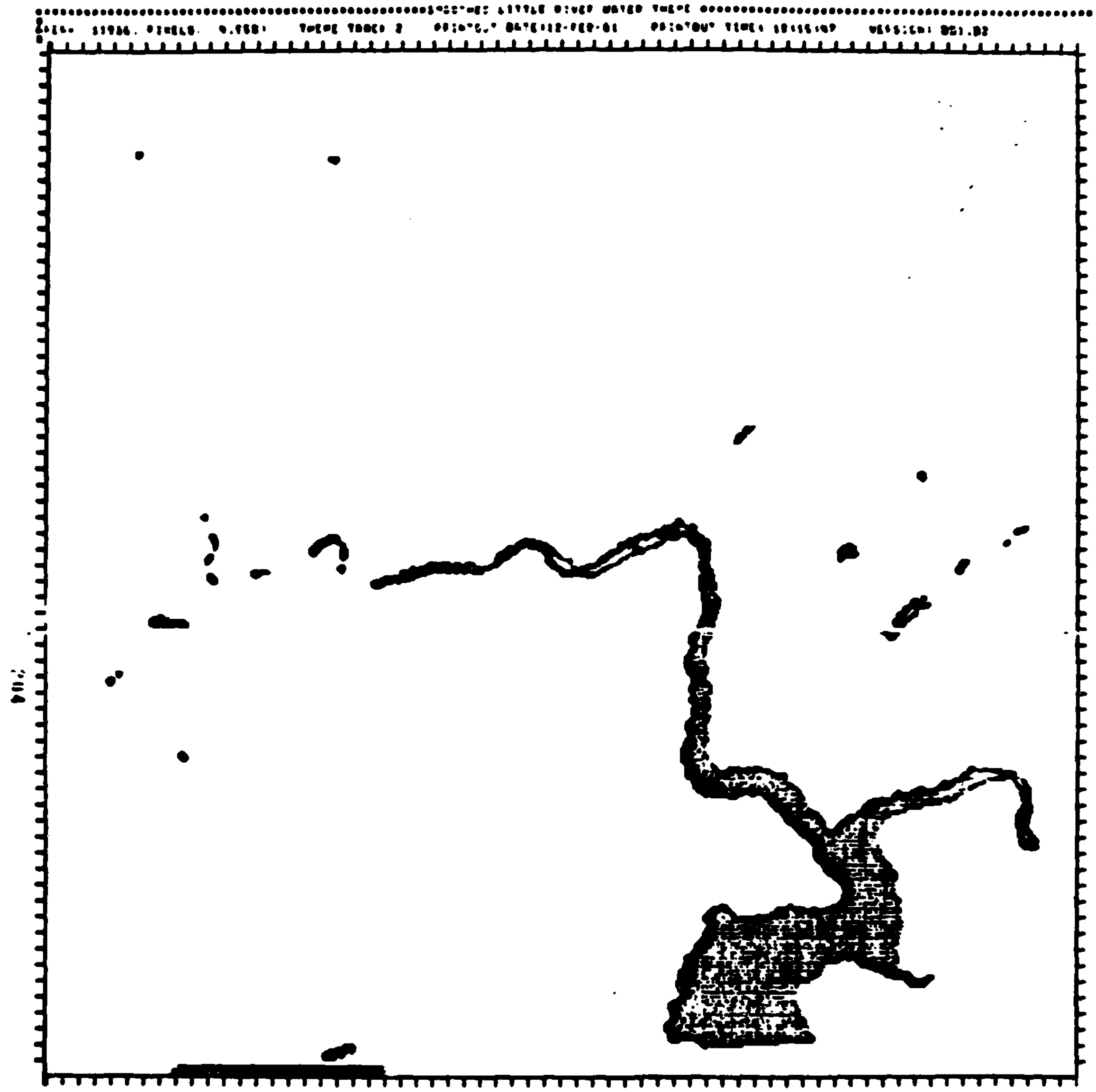


Figure 42. Smoothed Version of Figure 41.

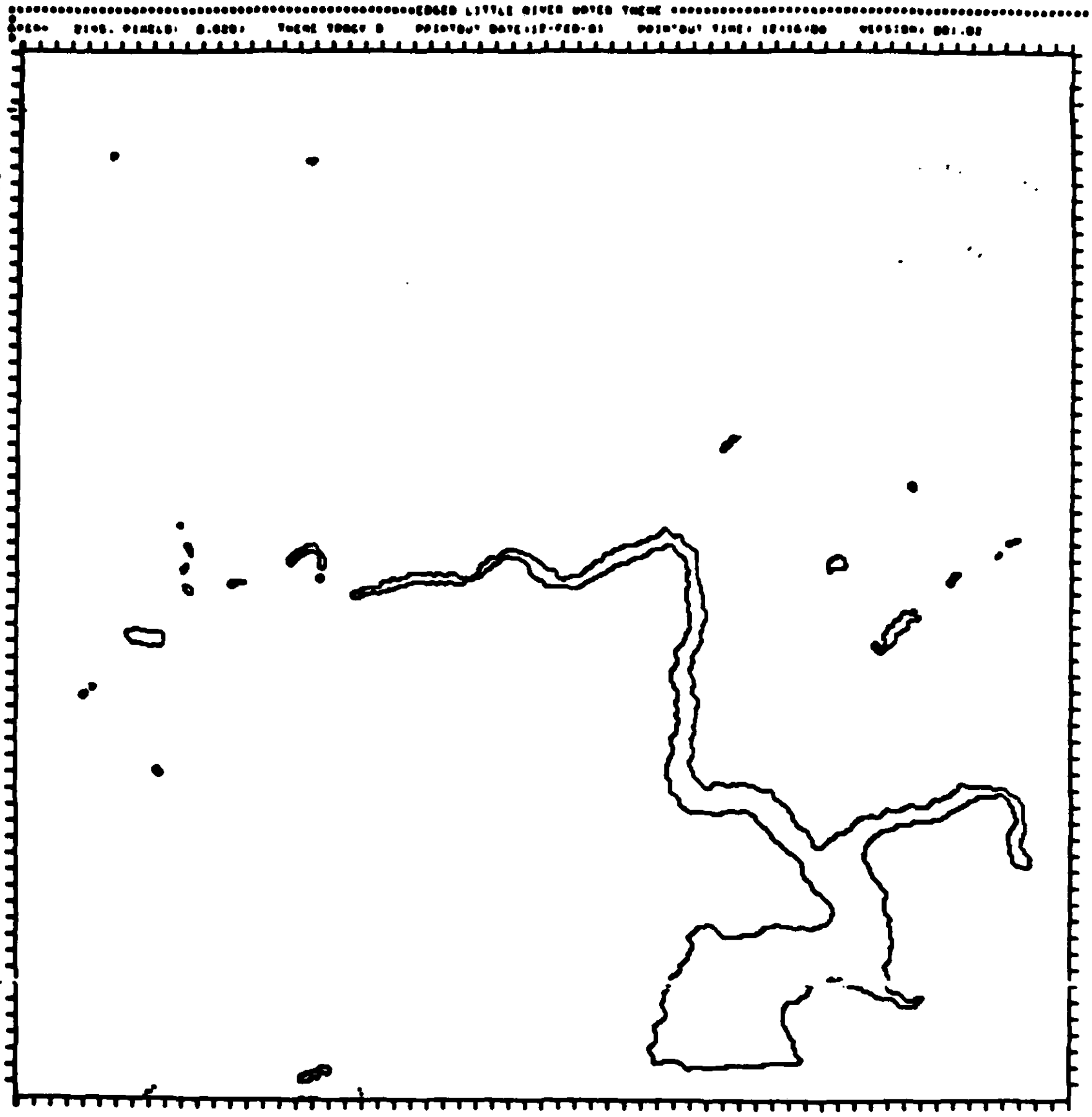


Figure 43. Edged Version of Figure 42

Figure 44. Chapel Creek Watercourse Theme Derived from Panchromatic Photography and SAR data.

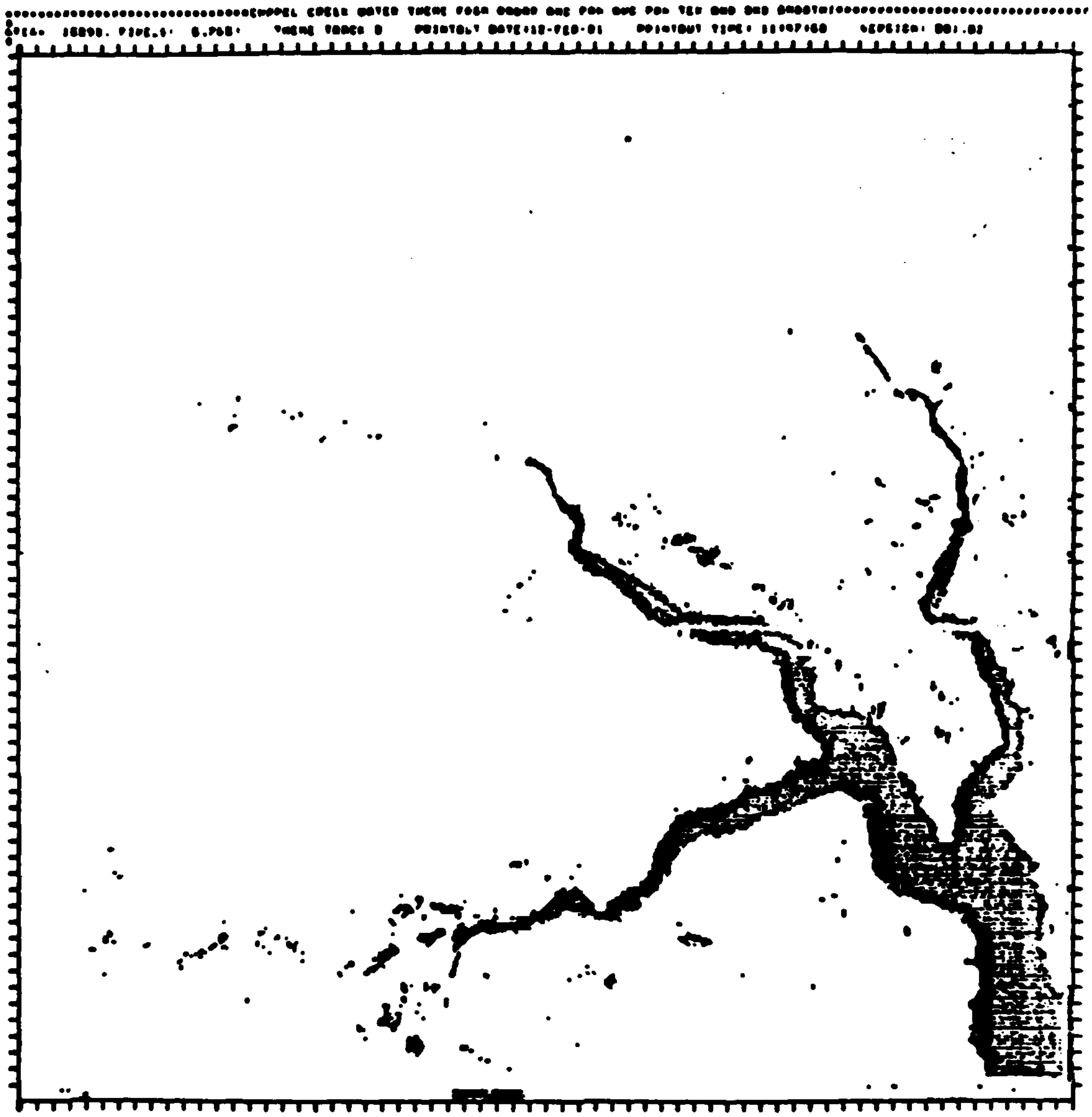


Figure 45. Chapel Creek Watercourse Theme Derived from Panchromatic Photography and SAR Data and from the Smoothed Microtexture Version of the Photography.

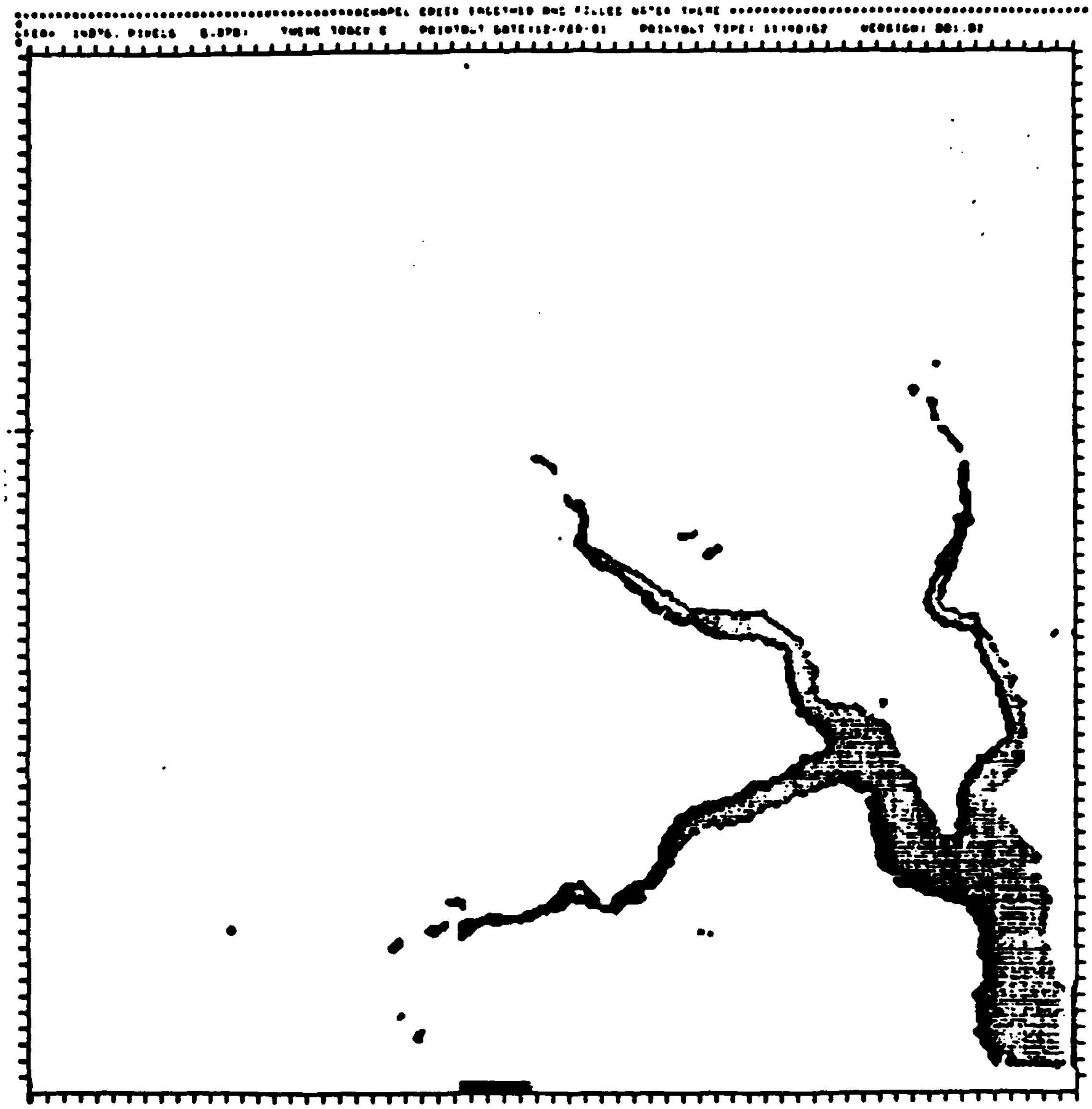


Figure 46. Smoothed Version of Figure 45.

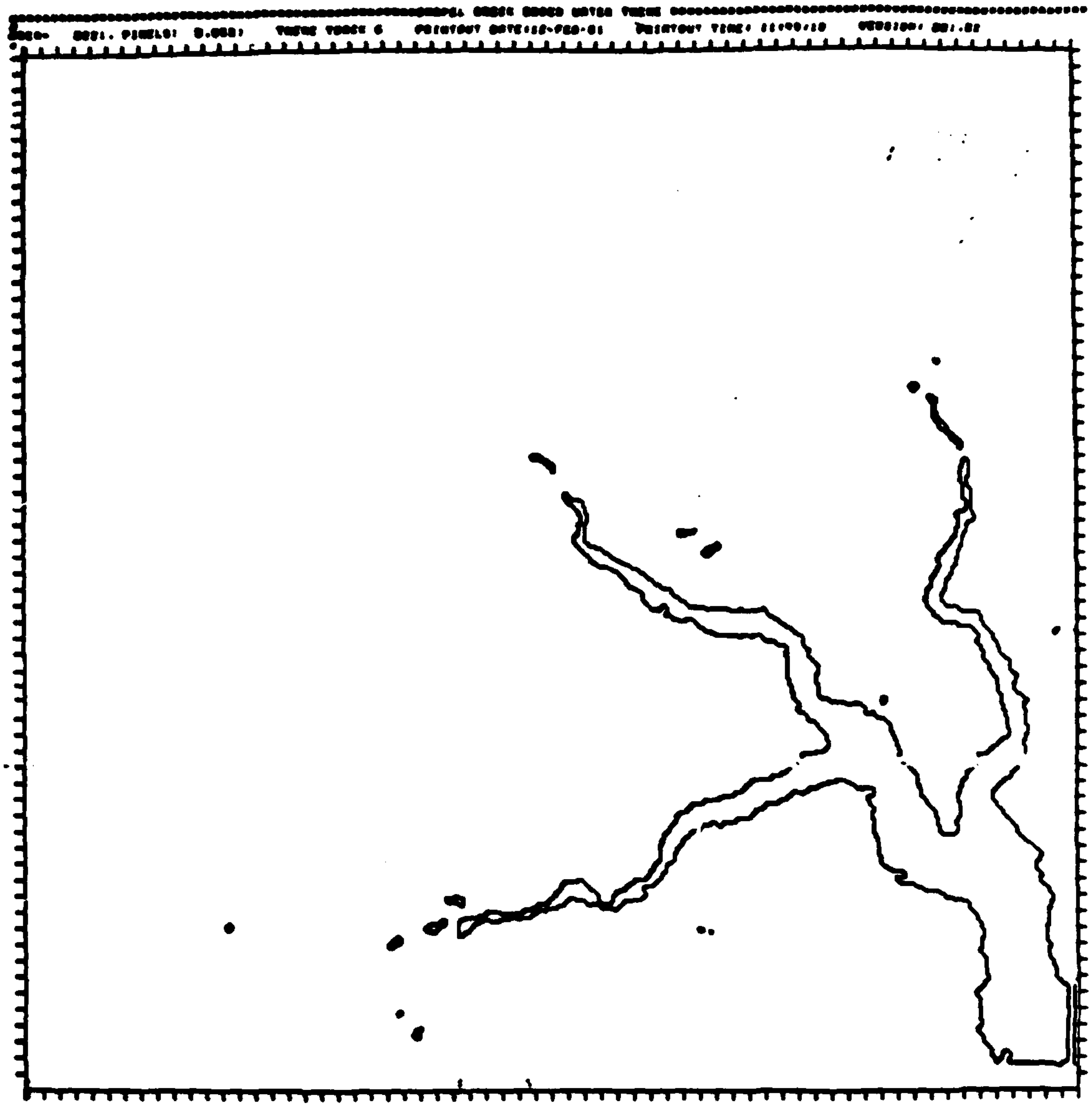


Figure 47. Edged Version of Figure 46.

4. Watercourse Extracting Using Panchromatic Imagery Digitized of Various Scales

A section of the panchromatic image of the Chapel Creek watercourse was digitized at multiple scales to determine what resolution of the derived micro-texture data contains the most separable watercourse class. Preliminary results indicate that panchromatic resolution between 1 and 2 meters yields microtexture data which best discriminates the watercourse class. This also, to some degree, verifies results for vegetation extraction where similar panchromatic resolutions yielded micro-texture data which contained best separability of vegetation classes.

- Alignment of all Water Bodies

  Given a suitable thematic classification of a watercourse, the alignment or centerline can be extracted by the methods detailed in Section III. Figure 12 shows the alignment of watercourse as extracted by the Golay method.

- Shore Alignment of all Water Bodies

  This is a relatively easy task once water bodies have been categorized. In fact, the recommended procedure is tolerant of classification errors of the omission type and required only a mediocre water body theme. The theme is filtered as required to generate a contigious map by using the theme filter or a similar operation, and is then outlined by the Golay process. Figure 47 shows a sample alignment theme.

B. Extraction of Dry Gaps and Associated Elements

1. Dry Gaps

Much of the effort involved in producing watercourse factor overlays deals with dry gap width, which is defined as the distance between stream bank shoulders. These width measurements are not usually derivable from a single image. The Terrain Analysis Procedural Guide for Drainage and Water Resources lists two methods of determining dry gap i.e (1) use of topographic information to locate lines of high elevation gradient along the stream bed and (2) locate the shoulder by photo interpretation of stereo pair. While there are interesting possibilities for implementing either of these approaches on an interactive system, no suitable imagery was available resulting in development efforts.

Without elevation information, the extraction of dry gaps is a task best left to the interpreter because of the large amount of judgement required. Dry gap outlines can be quickly entered into digital storage by using a pen/tablet or cursor in conjunction with an image display. Admittedly this is an inelegant solution, but it is also practical. Dry gaps can be located more easily on the basis of vegetation and land form interpretation, than by any unique machine extractable methods.

Once a watercourse and its associated dry gap have been satisfactorily mapped into themes by whatever means, interactive or automated, generation of factor overlays containing watercourse delineation and categorization still entails a series of tasks which are labor intensive and fairly tedious. Tasks which could be done effectively by computer include, the classification of water course segments, dry gap width, encoding of water course delineations by class, and the location of terminal points.

Even with the unfeasability of extracting dry gaps from the available imagery, it was felt that interactive computer methods could be of value in the extraction of these elements rather than not investigate the elements, some analysis was performed with a manually extracted dry gap as a starting point.

To demonstrate some of the potential tasks, a panchromatic image of Tennessee was used. The scene contained an unusually diverse set of watercourses with dry gap widths ranging from less than 3 to greater than 50 meters. Both the watercourse outline and dry gap outline were classified by manual means with an operator moving a joystick controlled cursor over the image.

While the resulting dry gap theme probably contains errors, it is typical of what can be done on the basis of land form and vegetation analysis with no available elevation data. Figure 48 and 49 show the results

2. <u>Categorizing Watercourse by Dry Gap Width</u>

A sequence which could lend itself to automated processing can be used with watercourse and dry gap themes stored. The first step is to employ a pixel stripping algorithm such as the Golay processor to locate the center line (delineation) of both themes. Figure 50 and 51 show the results

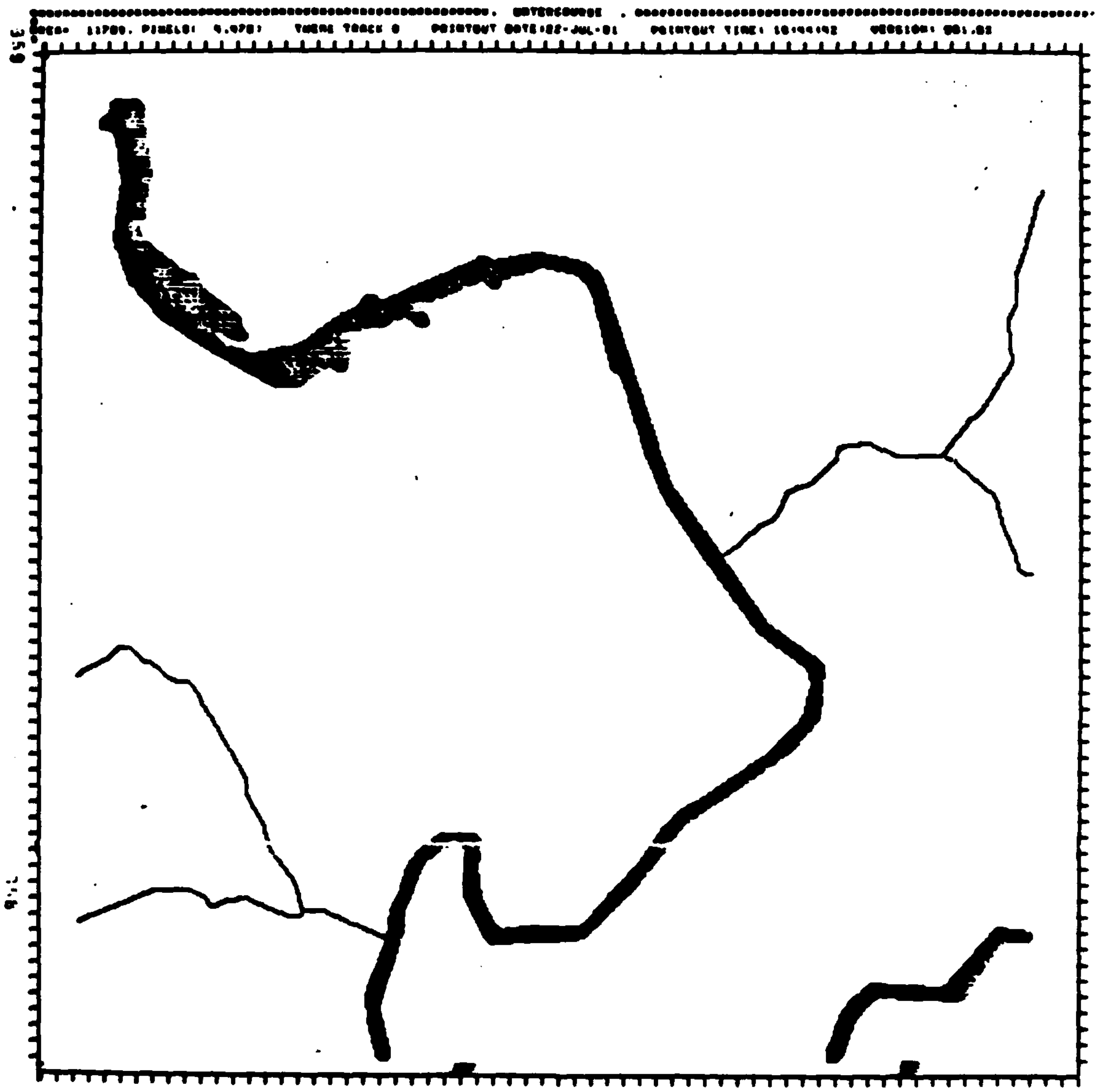


Figure 48. Manually Interpreted Watercourse

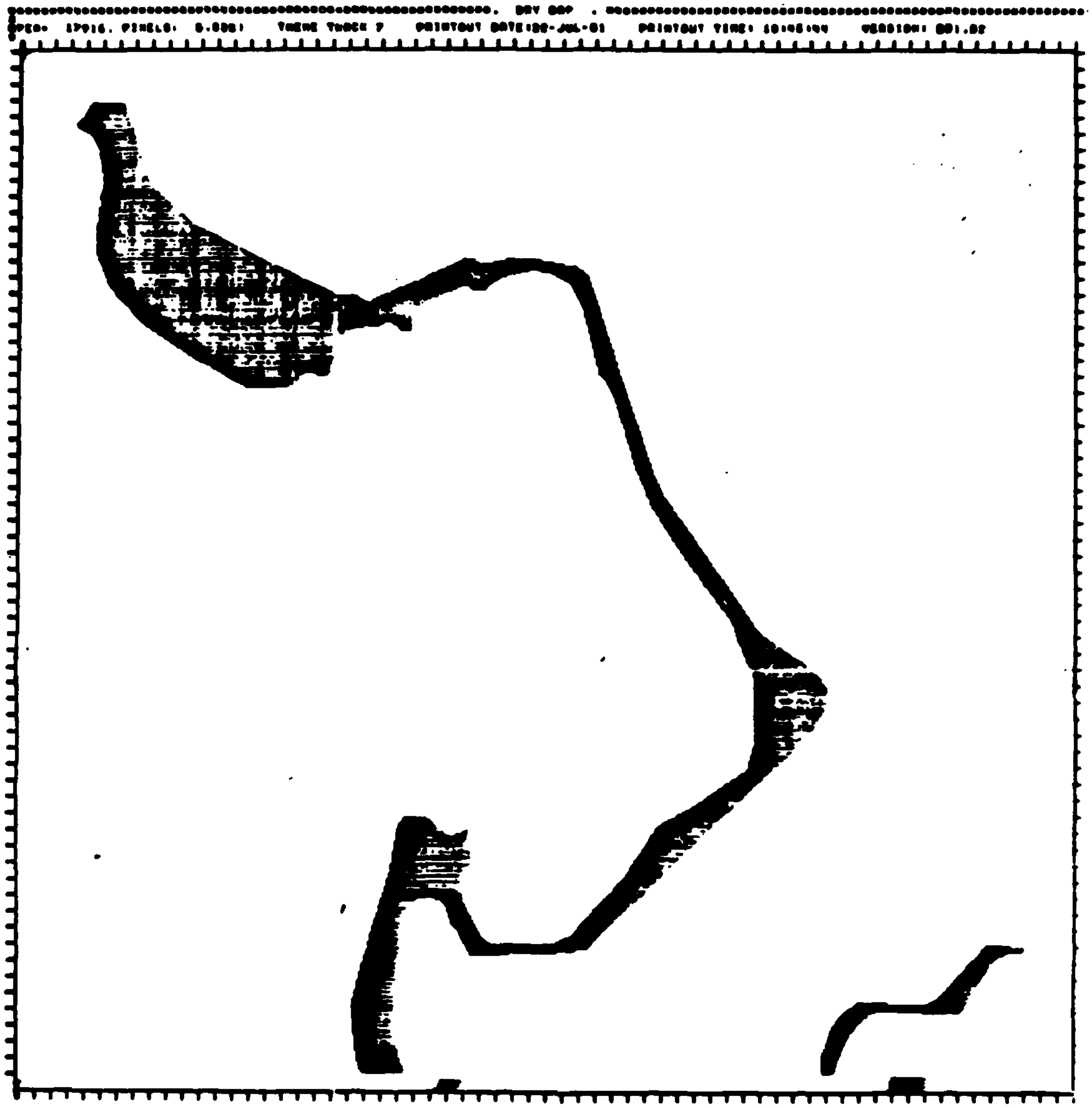


Figure 49. Manually Interpreted Dry Gap

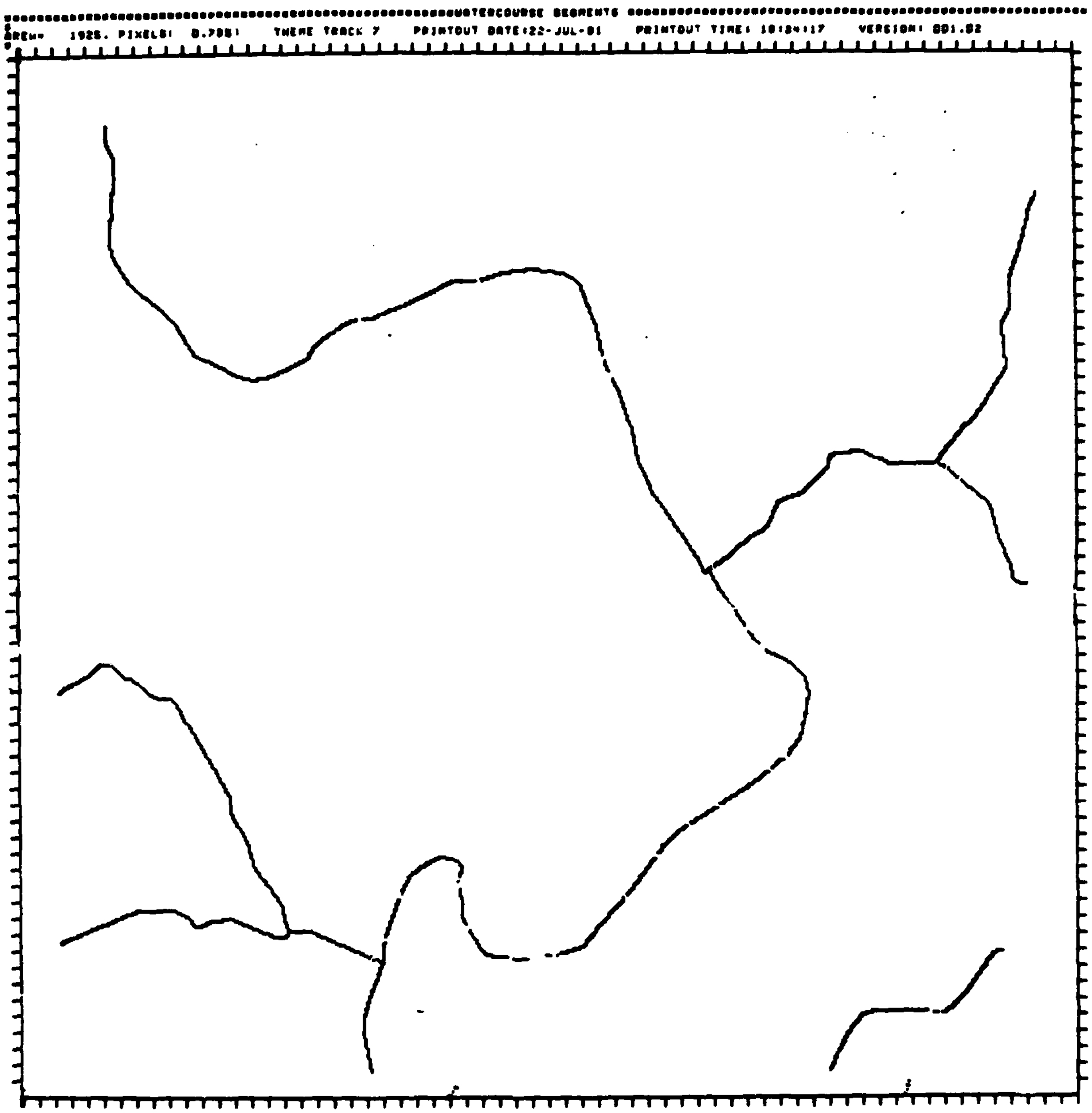


Figure 50. Skeletorized Watercourse

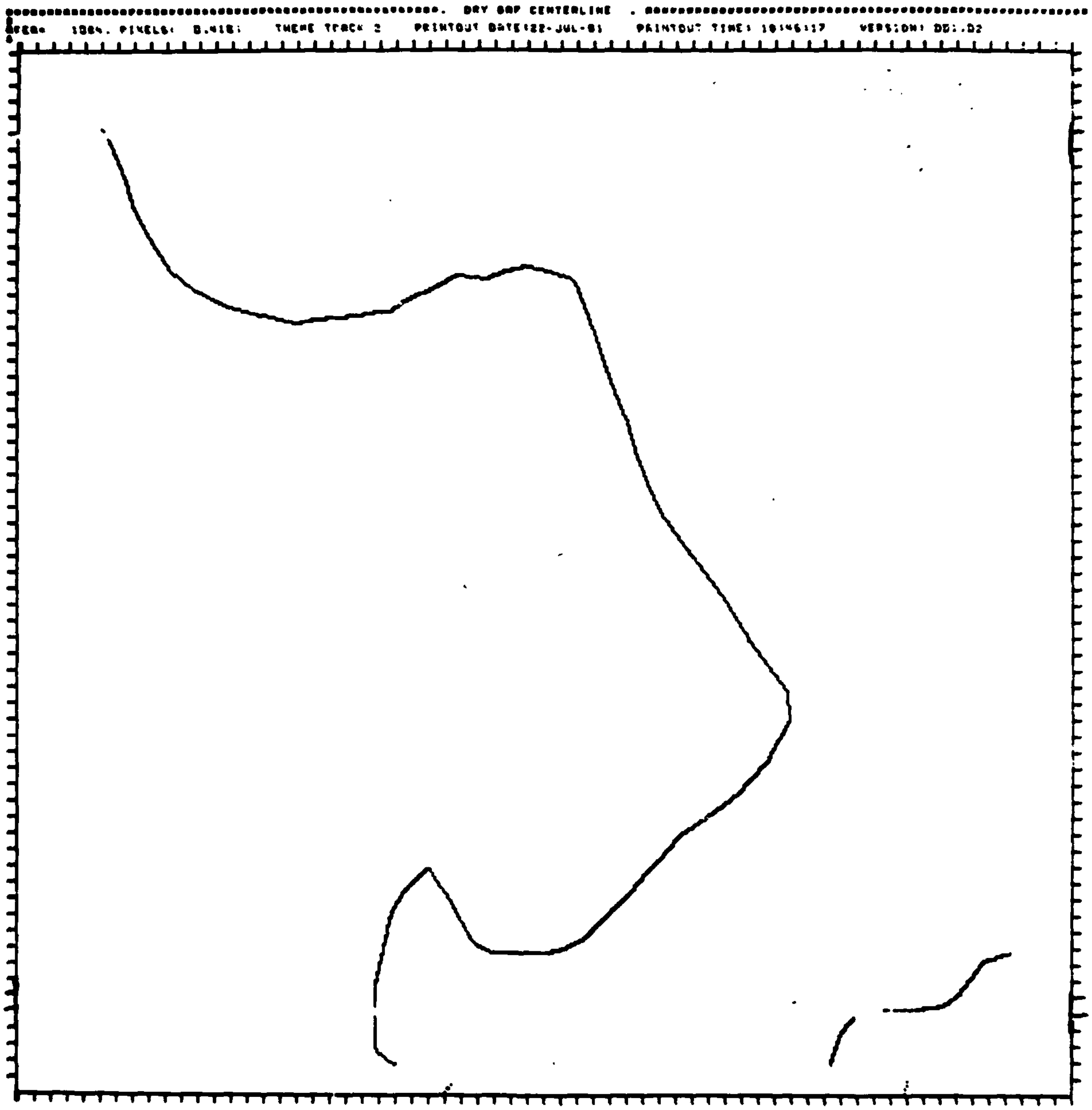


Figure 51. Skeletorized Dry Gap

for this particular image. These results have been touched up to correct some of the previously discussed problems introduced by the Golay skeletonizing algorithm as it is currently implemented on GE DIAL. A correction is needed to the algorithm which assures skeletonizing to a single pixel width line with no loss of connectivity.

The next step basically measures dry gap width along the entire water course by reapplying the Golay process to strip from the dry gap theme, annular rings of pixels corresponding to the desired width increments. In contrast to the previous process, the dry gap is not skeletonized but is allowed to disappear as the stripping continues. Five sequential applications of stripping can be used to yield a set of themes showing areas of dry gap greater than 3, 10, 18, 25 and 35 meters.

These themes are then logically combined with the dry gap center line producing a series of center line segments where each segment maps a dry width interval. Figure 52 is a flow chart describing the algorithm segments.

The desired final result is, of course, not a categorized dry gap center line, but is a categorized watercourse center line. The final steps of the sequence address this problem. A new set of themes must be generated, which categorize the entire area covered by the dry gap into segments by width interval. This is accomplished by applying the theme filter to the dry gap and its center line segments. Filter constraints are set up so that the dry gap theme is totally replaced with the theme corresponding to the closest center line segment. Finally, this set of themes is logically combined with the watercourse center line (or boundaries) to produce the desired categorization. See Figures 53 and 54.

3. Terminal Points of Watercourse Segments

At this point in the suggested processing sequence, we have suposedly performed a successful categorization of the watercourse based upon its associated dry gap width. The result is a series of themes, each containing segments of the watercourse delineation (centerline) corresponding to a given dry gap width increment. From any of these themes it is a simple matter to locate end points of the segments.

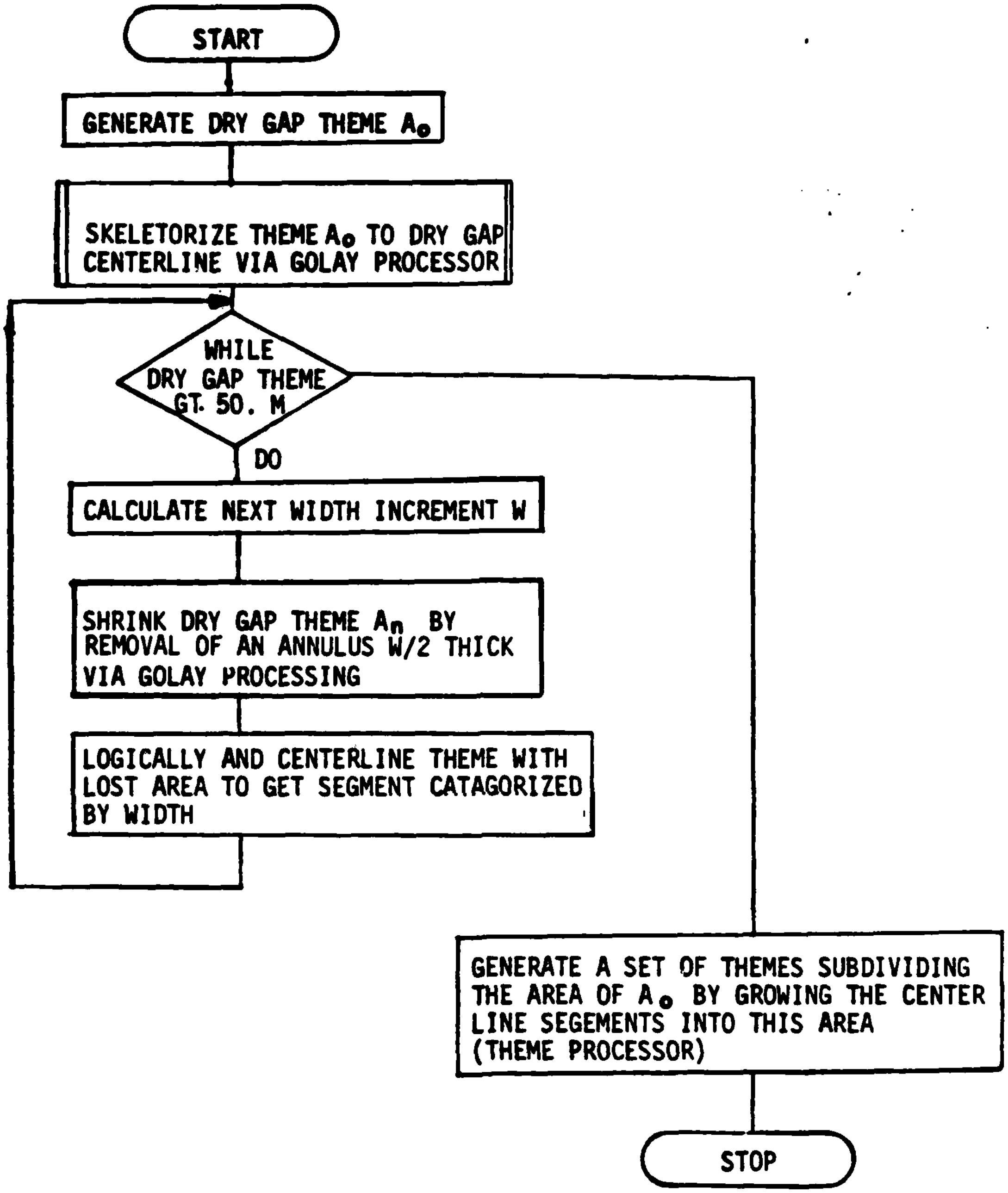


Figure 52. Categorizing a Dry Gap By Width

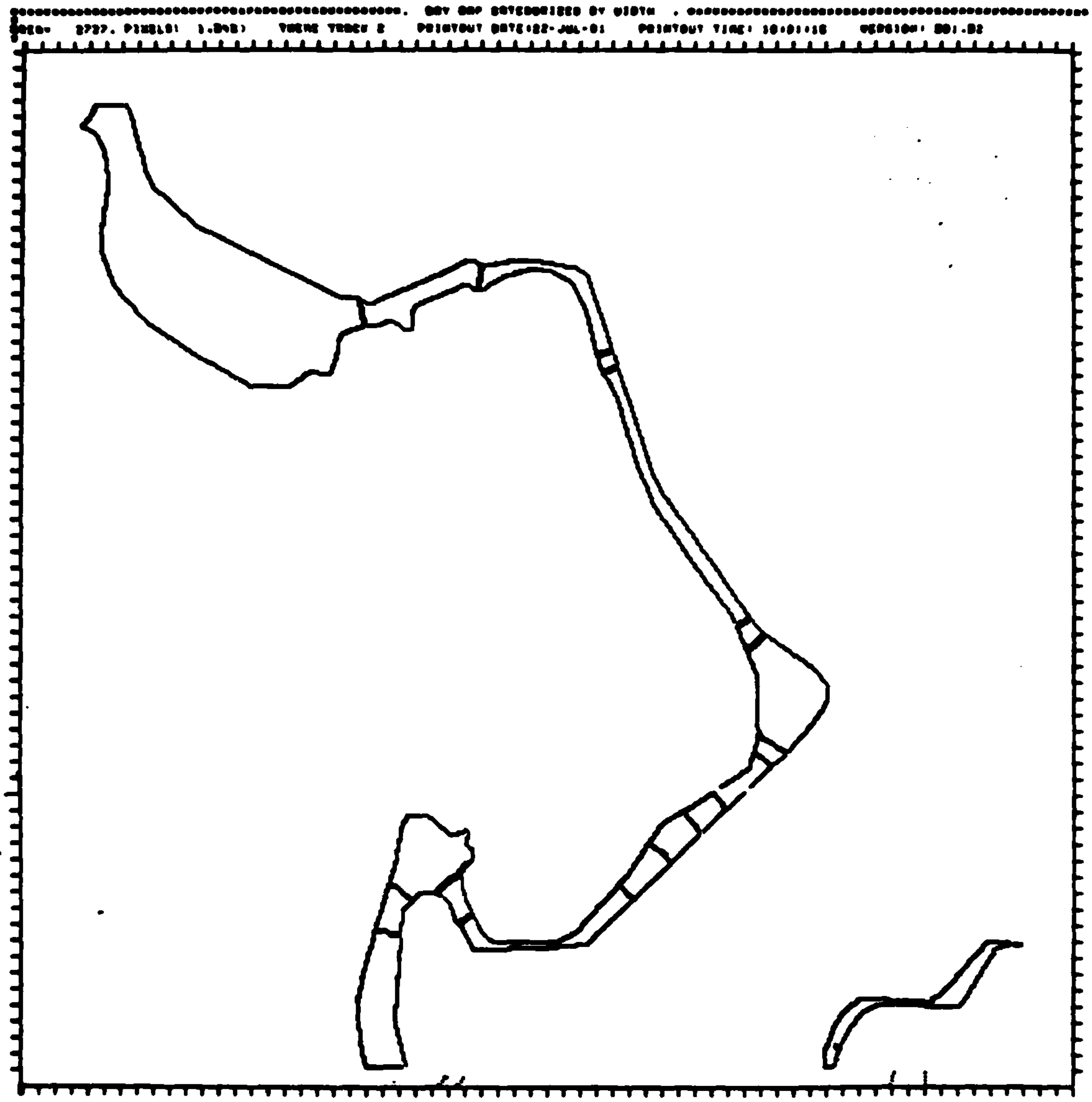


Figure 53. Dry Gap Categorized by Width and Displayed as Polygons

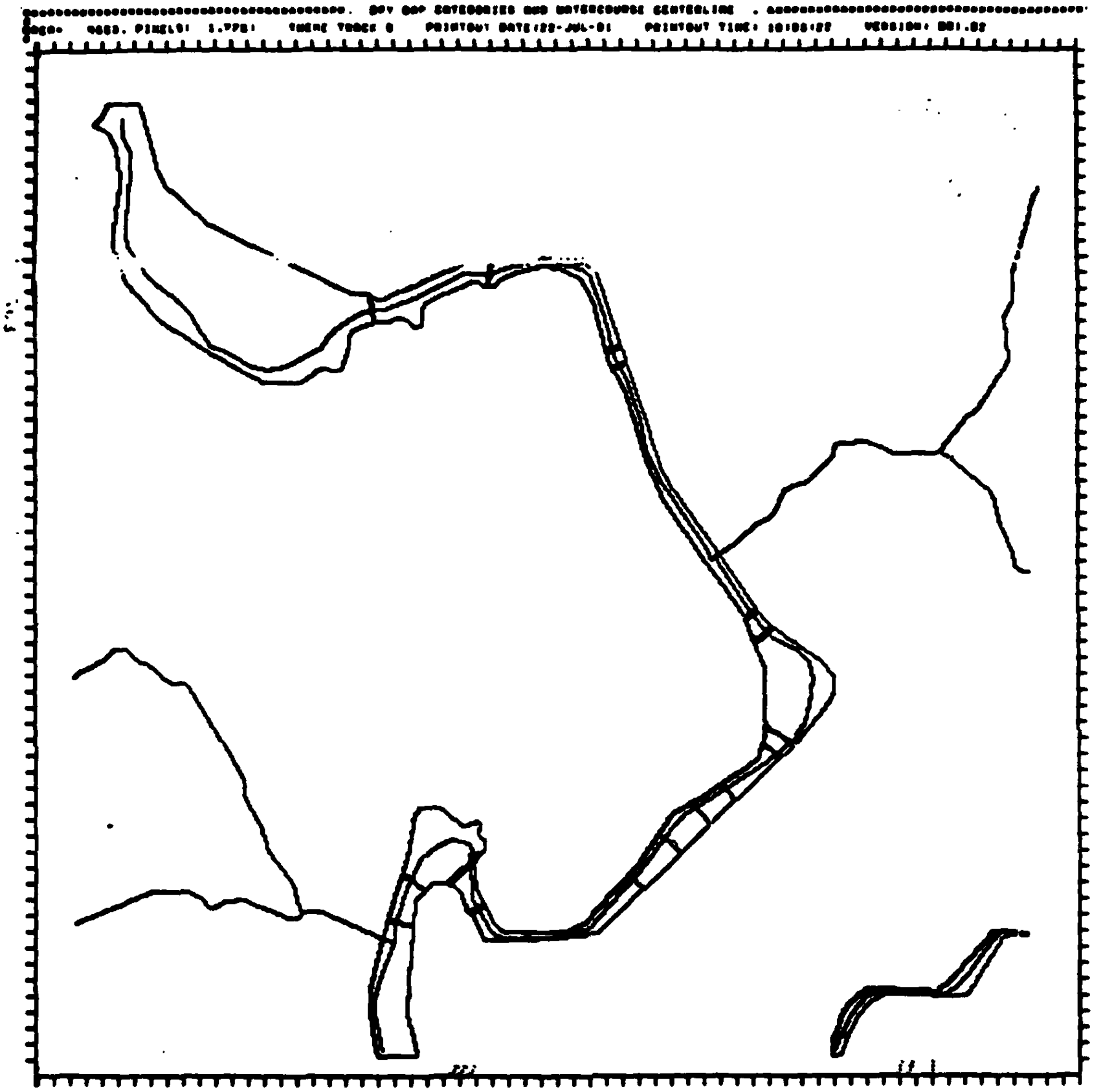


Figure 54. Combining Watercourse Center Line With Dry Gap Categories. Performing a Logical Intersection Produces Watercourse Segments Categorized Width.

Golay processing is a very straight forward means of finding segment end points. The procedure is to operate on the desired width increment theme by deleting all pixels with surrounds containing more than one pixel. Using the numbering convention in Figure 10, surrounds 2 thru 4 would be set for deletion. The output theme contains only the end points of connected segments in the input theme. The operation is completed in one pass over the image and is independent of the number of segments being processed.

For the end points of watercourse segments with less than three meter dry gaps an additional operation may be desirable. In this case, both terminal points of the segments are not used in factor overlays, only the points where the watercourse transitions to greater than three meter dry gap width. The same Golay operation applied to the entire watercourse delineation serves to locate points where the stream disappears. Subtracting these stream end points from the three meter segment end point leaves only the desired transitional points.

4. Encoding Watercourse Delineation

After categorization, the encoding of watercourses by line dashing, can and should be done by automated means. All the dry gap width symbols specified in the Terrain Analysis Procedural Guide for Drainage and Water Resources can be generated from a series of points equally spaced along the watercourse center line. Figure 55 is an algorithm (untested) which could be used to generate these points and Figure 56 lists methods by which the dot pattern could be used to encode lines with the desired patterns. Note that the feature encoding procedures are all raster (line by line) algorithms as opposed to line following algorithms. The raster processing approach has the advantages of neither being affected by image size, complexity, nor an occasional broken or non contiguous watercourse.

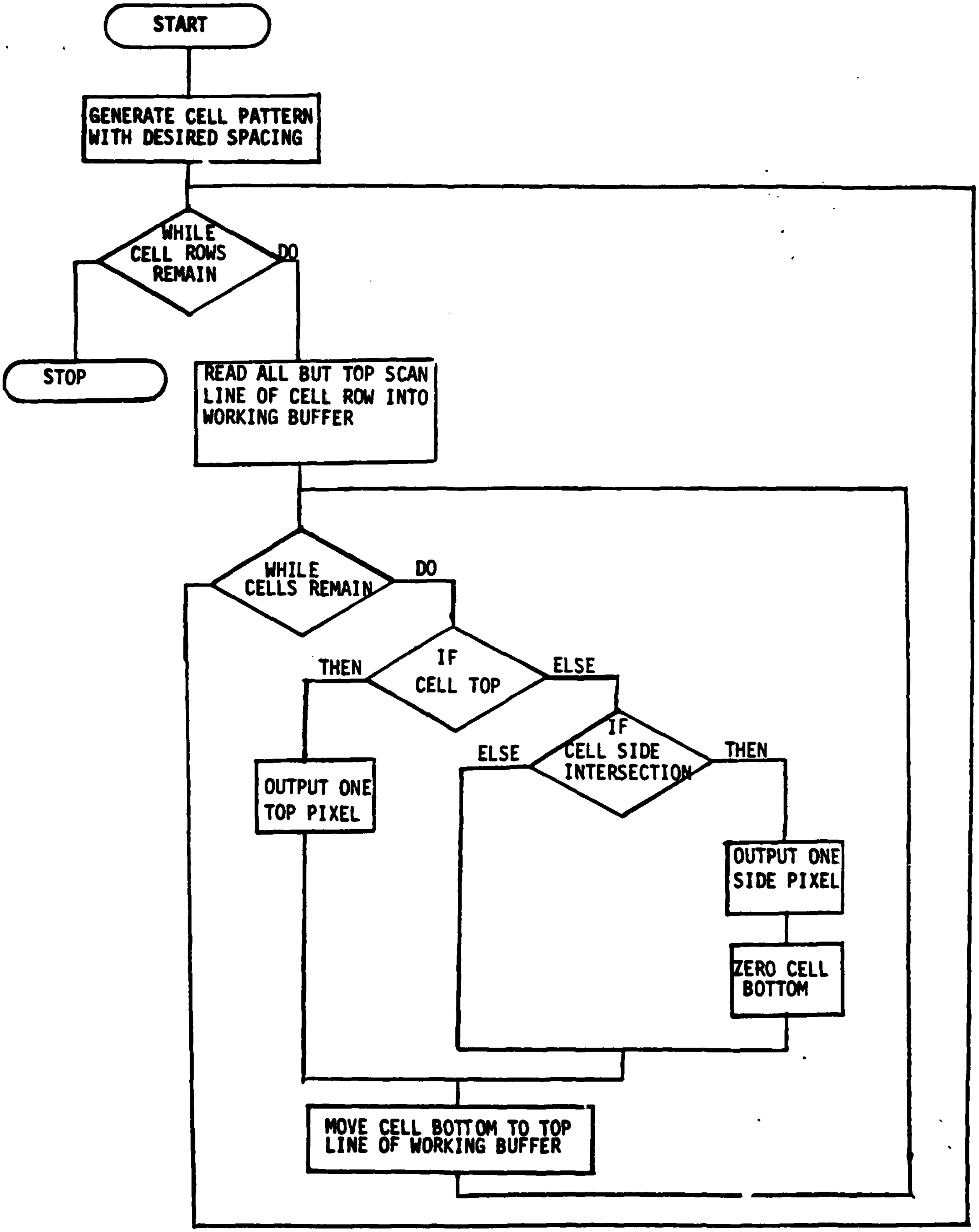


Figure 55. Raster Operating Line Dotting Algorithm

DRY GAP WIDTH (meters)	ENCODING PROCEDURE	RESULT
W<3	1. GENERATE CLOSELY SPACED DOTS FROM SINGLE PIXEL WIDTH WATER COURSE CENTER LINE. 2. GROW DOTS INTO SMALL CIRCULAR AREAS VIA GOLAY PROCESSING	. . . . ● ● ● ● ● ● ●
3<W<10	1. GENERATE WIDELY SPACED DOTS FROM CENTER LINE 2. GROW DOTS INTO CIRCULAR AREAS VIA GOLAY PROCESSING 3. OUTLINE CIRCULAR AREAS VIA GOLAY PROCESSING 4. SUBTRACT OUTLINE FROM CENTER LINE	. . . ● ● ● 0 0 0 —— - —— - —— - ——
10<W<18	1. GENERATE MODERATELY SPACED DOTS FROM CENTER LINE 2. GROW DOTS TO SMALL CIRCULAR AREAS 3. SUBTRACT AREAS FROM CENTER LINE	. . . . . ● ● ● ● ● — — — — —
18<W<25	1. USE UNMODIFIED CENTER LINE	————————
25<W<35	1. USE PROCEDURE FOR 10<W<18 2. GROW DASHED LINE VIA GOLAY PROCESSING	— — — — ▬ ▬ ▬ ▬
35<W<50	1. GROW CENTER LINE VIA GOLAY PROCESSING	████████

Figure 56. Encoding Watercourse Delineation

V. CONCLUSIONS

Phase 1 and Phase 2 of the study have concentrated on the feasibility and use of man-machine interactive digital processing techniques for the extraction of terrain data elements from aerial imagery. Interactive digital techniques were developed for vegetation/land cover boundary extraction and for extraction of forest-related data elements. These techniques included the use of spatial crown and texture operators.

Applications of these operators during Phase 1, enabled the extraction of feature roughness at several scales; extraction of vegetation types, wet areas and water bodies; and the determination of forested areas, forest canopy and locations of individual trees. The overall success of Phase 1 suggested the need to further test the operators relative to water resources data element and to improve and test the spatial crown and texture operators at different scales.

The development and coding of software included programs in image resolution and scale changing, feature skeletonizing, and theme smoothing. These programs are implemented on the GE DIAL system, primarily in Fortran language. Limited testing of the programs have been completed in Phase 2 to demonstrate usefulness. The implementation of these programs on systems similar to GE DIAL, is feasible.

Extraction of water resources data elements requires outlining features such as vegetation boundaries and shore line delineation of water bodies. Through spatial binary processing, or Golay processing, features were quickly and flawlessly outlined.

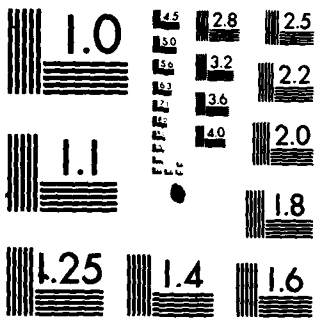
A slight variation of outlining, line stripping. provided a feasible, but somewhat lengthy (multiple pass) mean of categorizing watercourses by the width of their manually mapped dry gaps. Terminal points of watercourse segments can also be located by one pass Golay processing. Efforts to delineate watercourses, the prime water resources element, were only partially successful. The seemingly simple problem was to connect separated pixels of a watercourse theme, and then shrink or skeletonize the feature into a single pixel width centerline. Satisfactory delineation was achieved for certain cases but no universally applicable techniques were found.

AD-A114 815   GENERAL ELECTRIC CO LANHAM MD SPACE SYSTEMS DIV   F/G 9/2
INTERACTIVE DIGITAL IMAGE PROCESSING FOR TERRAIN DATA EXTRACTIO--ETC(U)
SEP 81   T F WESCOTT, W C DALLAM, C J PETERSON   DAAK70-79-C-0153

UNCLASSIFIED   ETL-0277   NL

2 of 2

END
DATE
FILMED
6-82
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

The production of usable factor overlays, which are the desired result of terrain element extraction was addressed and handled by writing what is thought to be an important theme filtering program. Its basic purpose was to perform spatial low pass filtering on a categorized image as required to generate overlays which resemble those made by photointerpretation and which are spatially simple enough to be combined into complex overlays. Provisions are made to restrict change of critical or exceptionally accurate themes and to remap on a spatial basis those which are incidental or plagued with categorization errors.

VI. RECOMMENDATIONS

In the performance of Phase 2, areas were identified as candidates for further investigation. Generally, these candidates include work described as testing of existing software/techniques and development of new software/techniques. The recommended areas for future work are:

- Continue to perform sufficient testing of techniques developed in both Phase 1 and Phase 2. The testing should include various sites and ground data to radiate the techniques.
- Limited results were obtained for delineation of line features, i.e., streams and roads. It is believed that the existing Golay algorithm, an already invaluable tool, could be revised and made to perform error free skeletonizing.
- Further testing of the theme filtering program is suggested. This testing is relative to the production of factor overlap, i.e. a comparison of manual interpretation vs. computer. The theme filtering program has proven valuable as a cosmetic noise cleaner and as a tool in the feature extraction process. It allows satisfactory results to be filtered from otherwise unacceptable themes. Although not tested in this respect, it is also seen as a promising means of digitally combining selected themes into complex overlays. While highly effective, the program is somewhat cumbersome to set up, and the operator interface should be rewritten so that commonly used filtering options are menu selectable. With the minor change the program is recommended for general interactive use.
- To date, processing of imagery data has been restricted to small geographical areas. The testing of developed techniques from Phase 1 and 2 to mosaicked images is needed relative to the practical uses of terrain analysis. It is recommended that such work be performed to allow for the expansion of the techniques to largen geographical areas.

APPENDIX A

IMAGE RESOLUTION AND SCALE CHANGING PROGRAM

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C               IMAGE SCALE AND RESOLUTION CHANGING
C
C     THIS PROGRAM PROVIDES A MEANS OF MODIFYING IMAGE RESOLUTION
C     TO SIMULATE ANY SENSOR WITH LOWER RESOLUTION THAN THAT INHERRENT
C     IN THE ORIGINAL IMAGE DATA, AND TO PERFORM GOEMETRIC SCALING.
C
C     ITS MAIN PURPOSE IS TO MODIFY IMAGES IN A MANNER THAT ALLOWS
C     STANDARD SPATIAL OPERATORS TO BE APPLIED TO ANY IMAGE AT A VARIETY
C     OF SCALES, THUS EXTRACTING SPATIAL INFORMATION AT A VARIETY OF
C     FREQUENCIES WITH MINIMUM COMPUTATION.  THE SUGGESTED PROCESSING SEQUENCE
C     IS TO USE THIS PROGRAM TO DEGRADE RESOLUTION TO THE POINT WHERE A
C     DESIRED FEATURE DISPLAYS UNIQUE SPATIAL PROPERTIES, APPLY A STANDARD
C     OPERATOR, AND THEN RESCALE THE OUTPUT BACK TO ITS ORIGINAL SIZE FOR
C     SUBSEQUENT FEATURE EXTRACTION.
C
C
C     FORTRAN PROGRAM MODULES:
C
C       BGWNDO ---A DRIVER THAT CALLS THE MODULES NECESSARY TO PERFORM
C                 ANY SELECTED SCALING OPERATION.
C
C       WNDO   ---PROVIDES AN INTERACTIVE METHOD OF DEFINING INPUT AND OUTPUT
C                 IMAGE SEGMENTS.  MAKES USE OF THE IMAGE-100 JOYSTICK CONTROLLED
C                 RECTANGULAR CURSOR.
C
C       SELECT ---ACCEPTS OPERATOR SELECTIONS  OF INPUT AND OUTPUT IMAGE PLANES
C                 OR CHANNELS IN THE IMAGE 100 REFRESH MEMORY.
C
C       NNWNDO ---PERFORMS IMAGE MAPPING VIA NEAREST NEIGHBOR INTERPOLATION.
C
C       AVWNDO ---PERFORMES IMAGE MAPPING VIA COMPUTATION OF LOCAL AREAL
C                 AVERAGES.  USED TO SMOOTH OR SHRINK IMAGES.
C
C       CDWNDO ---PERFORMES IMAGE MAPPING VIA A SUBPIXEL CONTEXT DEPENDANT
C                 INTERPOLATION.  USED TO ENLARGE IMAGES.
C
C     PDP-11 ASSEMBLER MODULES:
C
C       LNSCAL ---CALLED BY NNWNDO TO REMAP PIXELS ALONG A SCAN LINE
C
C       COLSUM ---CALLED BY AVWNDO TO MAINTAIN RUNNING SUMS ALONG IMAGE COLUMNS
C
C       ROWSUM ---CALLED BY AVWNDO TO  COMPUTE RUNNING AVERAGES ALONG IMAGE ROWS
C
C
C     IMAGE 100 LIBRARY SUBROUTINES:
C
C       FRONT
C       INTFF  ---FREE FORMAT INPUT ROUTINES
C
C       IPK
C       IWK    ---CURSOR CONTROL ROUTINES
C
C       IPV
```

```
FORTRAN IV      V01C-03        THU 12-NOV-81 14:08:49        PAGE 002
CORE=11K, UIC=[127,4]                      DB:HGWNDO,LP:=DB:DGWNDO
      C      IVV    ---VIDEO MEMORY ACCESS ROUTINES
      C
      C      OUTPUT ---TERMINAL CONTROL ROUTINE
      C
      C
      C     KEY VARIABLES:
      C
      C      IX1,IY1,IX2,IY2  ---COORDINATES OF UPPER LEFT AND LOWER RIGHT CORNERS OF
      C                          THE INPUT IMAGE WINDOW.
      C
      C      OX1,OY1,OX2,OY2  ---COORDINATES OF UPPER LEFT AND LOWER RIGHT CORNERS
      C                          OF THE OUTPUT IMAGE.
      C
      C      INCHS(5),OUCHS(5)---CHANNEL ASSIGNMENT ARRAYS.  INCHS(N) GETS PROCESSED
      C                          ONTO OUCHS(N).
      C
      C      INSERT           ---A FLAG THAT DETERMINES IF THE OUTPUT IMAGE IS TO
      C                          TOTALLY REPLACE PREVIOUS OUTPUT CHANNEL CONTENTS, OR
      C                          IS TO BE INSERTED INTO A WINDOW AREA
      C
      C      RMAG             ---MAGNIFICATION FACTOR
      C
      C      BUF(40)          ---ALPHA NUMERIC STRING FOR OPERATOR INPUTS
      C
      C      DX(512), DY(512) ---GEOMETRIC MAPPING TABLES.  TABLE INDEX REFERS TO
      C                          OUTPUT PIXEL LOCATIONS, AND TABLE CONTENTS REFER
      C                          TO CORRESPONDING INPUT PIXEL LOCATIONS.
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0001         END
```

A2

```
FORTRAN IV      V01C-03         THU 12-NOV-81 14:08:57          PAGE 001
CORE=11K, UIC=[127,4]                          DB:BGWNDO,LP:=DB:BGWNDO

      C
0001          SUBROUTINE BGWNDO
      C
      CCCCCCCCCCCCCCCCCCCCCC GENERAL * ELECTRIC CCCCCCCCCCCCCCCCCCCCCCCC
      C
      C       GENERAL CHANNEL TO CHANNEL WINDOWING PROGRAM
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC TFW 11/80
      C
0002          IMPLICIT INTEGER(A-Q,S-Z)
      C
0003          DIMENSION BUF(40)
      C
0004          LOGICAL INSERT
      C
0005          COMMON /WINDO/ IX1,IY1,IX2,IY2,OX1,OY1,OX2,OY2,RMAG,INSERT
0006          COMMON /CHANL/ INCHS(5), OUCHS(5)
      C
0007          DATA UTY /6/
0008          DATA N/'N'/, Y/'Y'/
0009          DATA I/'I'/, C/'C'/, IR/'R'/
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
      C       CLEAR DISPLAY
      C
0010          CALL OUTPUT(27,12,7)
      C
      C       DETERMINE WINDOW MAPPING
      C
0011          CALL WINDOW
0012          CALL OUTPUT(27,12,7)
0013          WRITE(UTY,999)IX1,IY1,IX2,IY2,OX1,OY1,OX2,OY2,RMAG
0014    999   FORMAT(/' MOVE INPUT  (',I3,',',I3,'), (',I3,',',I3,')'/      report window
         &           ' TO OUTPUT  (',I3,',',I3,'), (',I3,',',I3,')'/      corner locations
         $           ' MAGNIFICATION = ',F5.2,/)
      C
      C       SELECT CHANNELS
      C
0015          CALL SELECT(INCHS,OUCHS)
      C
      C       ASK FOR MAPPING TECHNIQUE
      C
0016          TECHNI=N
0017          IF(RMAG.EQ.1.0)  GO TO 120                                        prompt for desired
0019    101      WRITE(UTY,1010)                                                resampling techniques
0020    1010     FORMAT(/'$(C)OMPUTED OR (N)EAREST NEIGHBOR INTERPOLATION >')
0021             READ(UTY,1020)BUF
0022    1020     FORMAT(40A2)
0023             CALL FRONT(BUF)
0024             IF((BUF(1).NE.N).AND.(BUF(1).NE.C)) GO TO 101
0026             TECHNI=BUF(1)
0027    120   CONTINUE
      C
      C       ASK FOR OUTPUT METHOD
      C
0028          DELTAX=OX2-OX1
0029          DELTAY=OY2-OY1
0030          INSERT=.FALSE.
```

581

```
FORTRAN IV      V01C-03        THU 12-NOV-81 14:08:57        PAGE 002
[illegible] UIC=[127,4]                   DB:BGWNDO,LP:=DB:BGWNDO

      C
            IF((DELTAX.GT.510).AND.(DELTAY.GT.360)) GO TO 130
        129    WRITE(UTY,1290)
        1290   FORMAT(/'$(I)NSERT OR (R)EPLACE OUTPUT >'')
               READ(UTY,1020)BUF
               CALL FRONT(BUF)
               IF((BUF(1).NE.I).AND.(BUF(1).NE.IR)) GO TO 129
               IF(BUF(1).EQ.I) INSERT=.TRUE.
        130 CONTINUE
      C
      C     DETERMINE PROCESSING CASE
      C
      C     NEAREST NEIGHBOR CASE?
            IF((TECHNI.EQ.N).OR.(RMAG.EQ.1.0))        CALL NNWNDO
      C
      C     INTERPOLATION CASE?
            IF((TECHNI.EQ.C).AND.(RMAG.GT.1.0))       CALL CDWNDO
      C
      C     AVERAGING CASE?
            IF((TECHNI.EQ.C).AND.(RMAG.LT.1.0))       CALL AV.NDO
      C
            RETURN
            STOP
            END
```

prompt for
output method

call appropriate
resampling module

```
FORTRAN IV      V01C-03         THU 12-NOV-81 14:09:09          PAGE 001
???-11K. UIC=[127,4]                          DB:BGWNDO,LP:=DB:BGWNDO

      SUBROUTINE WNDOW
C
CCCCCCCCCCCCCCCCCCCCCC GENERAL * ELECTRIC CCCCCCCCCCCCCCCCCCCCCCCCC
C
C     PROCESSING WINDOW DEFINITION VIA CURSOR POSITIONING
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC TFW 11/80
C
      IMPLICIT INTEGER(A-O,S-Z)
C
      DIMENSION BUF(40)
      DIMENSION INCUR(5), OUTCUR(5)
C
      REAL FLOAT
C
      COMMON /WINDO/ IX1,IY1,IX2,IY2,OX1,OY1,OX2,OY2,RMAG,INSERT
C
      DATA UTY /6/
      DATA N/'N'/, Y/'Y'/
      DATA NULL/' '/
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ASK OPERATOR FOR MAGNIFICATION
C
 90   WRITE(UTY,9000)
 9000 FORMAT(//'$ENTER MAGNIFICATION (EG. 2:1) >')
 100  READ(UTY,1000)BUF
 1000 FORMAT(40A2)
C
      IF(BUF(1).EQ.X) CALL EXIT
      P=0
      CALL INTFF(P,BUF,GO,NUMER)
      CALL INTFF(P,BUF,EO,DENOM)
C
C     TAKE CARE OF INPUTS OF ZERO
      IF(NUMER.EQ.0) GO TO 90
      IF(DENOM.EQ.0) DENOM=1
      RMAG=(FLOAT(NUMER))/(FLOAT(DENOM))
C
C     LOCATE WINDOWS WITH CURSOR
C
                            IF(RMAG.LT.1.0) GO TO 300
C
C     CASE #1          WINDOW WILL BE ENLARGED
C
          ODX=512
          ODY=512
          OUTCUR(2)=256
          OUTCUR(3)=ODX/2
          OUTCUR(4)=256
          OUTCUR(5)=ODY/2
C
 201      WRITE(UTY,2010)ODX,ODY
 2010     FORMAT(/'$OUTPUT WINDOW = ',I3,' X ',I3,' >')
          READ(UTY,1000) BUF
          IF(BUF(1).EQ.X) CALL EXIT
          IF(BUF(1).EQ.NULL) GO TO 210
C             ELSE REPLACE FULL SIZE OUTPUT DEFAULTS
```

Note: This module must be replaced for non Image 100 implementation

```
0039                P=0
0040                CALL  INTFF(P,BUF,80,ODX)
0041                CALL  INTFF(P,BUF,80,ODY)
0042                IF((ODX.LE.0).OR.(ODX.GE.512)) GO TO 201
0044                IF((ODY.LE.0).OR.(ODY.GE.512)) GO TO 201
      C
      C             SIZE CURSOR FOR REDUCED OUTPUT WINDOW
      C
0046                OUTCUR(3)=ODX/2
0047                OUTCUR(5)=ODY/2
0048                CALL  IWK(OUTCUR)
0049   205          WRITE(UTY,2050)
0050   2050         FORMAT('$POSITION CURSOR FOR OUTPUT WINDOW, THEN CR >')
0051                READ(UTY,1000)NOTHIN
0052                CALL  IRK(OUTCUR)
0053   210      CONTINUE
      C
      C         SIZE CURSOR TO INPUT AREA
      C
0054            IDX=(FLOAT(ODX))/RMAG
0055            IDY=(FLOAT(ODY))/RMAG
0056            CALL  IRK(INCUR)
0057            INCUR(3)=IDX/2
0058            INCUR(5)=IDY/2
0059            CALL  IWK(INCUR)
0060   215      WRITE(UTY,2150)
0061   2150     FORMAT('$POSITION CURSOR FOR INPUT WINDOW, THEN CR >')
0062            READ(UTY,1000)NOTHIN
0063            CALL  IRK(INCUR)
0064            GO TO 400
0065   300      CONTINUE
      C
      C
      C
      C     CASE #2           WINDOW WILL BE REDUCED
      C
0066            IDX=512
0067            IDY=512
0068            INCUR(2)=256
0069            INCUR(3)=IDX/2
0070            INCUR(4)=256
0071            INCUR(5)=IDY/2
      C
0072   301      WRITE(UTY,3010)IDX,IDY
0073   3010     FORMAT(/'$INPUT WINDOW = ',I3,' X ',I3,' >')
0074            READ(UTY,1000) BUF
0075            IF(BUF(1).EQ.X) CALL EXIT
0077            IF(BUF(1).EQ.NULL) GO TO 310
      C             ELSE REPLACE DEFAULTS
0079                P=0
0080                CALL  INTFF(P,BUF,80,IDX)
0081                CALL  INTFF(P,BUF,80,IDY)
0082                IF((IDX.LE.0).OR.(IDX.GE.512)) GO TO 301
0084                IF((IDY.LE.0).OR.(IDY.GE.512)) GO TO 301
      C
      C             SIZE CURSOR FOR SMALLER INPUT WINDOW
      C
0086                INCUR(3)=IDX/2
0087                INCUR(5)=IDY/2
```

```
                 CALL IWK(INCUR)
   305           WRITE(UTY,3050)
   3050          FORMAT('$POSITION CURSOR FOR INPUT WINDOW, THEN CR >')
                 READ(UTY,1000)NOTHIN
                 CALL IRK(INCUR)
   310        CONTINUE
 C
 C            SIZE CURSOR TO OUTPUT AREA
 C
              ODX=(FLOAT(IDX))*RMAG
              ODY=(FLOAT(IDY))*RMAG
              CALL IRK(OUTCUR)
              OUTCUR(3)=ODX/2
              OUTCUR(5)=ODY/2
              CALL IWK(OUTCUR)
   315        WRITE(UTY,3150)
   3150       FORMAT('$POSITION CURSOR FOR OUTPUT WINDOW, THEN CR >')
              READ(UTY,1000)NOTHIN
              CALL IRK(OUTCUR)
   400     CONTINUE
 C
 C         SET WINDDOWS IN COMMON
 C
           IX1=INCUR(2)-IDX/2
           IY1=INCUR(4)-IDY/2
           IX2=IX1+IDX
           IY2=IY1+IDY
 C
           OX1=OUTCUR(2)-ODX/2
           OY1=OUTCUR(4)-ODY/2
           OX2=OX1+ODX
           OY2=OY1+ODY
 C
           RETURN
           END
```

```
FORTRAN IV      V01C-03           THU 12-NOV-81 14:09:23          PAGE 001
CORE 11K, UIC=[127,4]                       DB:BGWNDO,LP:=DB:BGWNDO

0001          SUBROUTINE SELECT(INCHS,OUTCHS)
        C
        CCCCCCCCCCCCCCCCCCCCC GENERAL * ELECTRIC CCCCCCCCCCCCCCCCCCCCCCCCC
        C
        C     SETS UP ARRAYS OF INPUT AND TARGET CHANNELS
        C
        CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC TFW 11/80
        C
0002          IMPLICIT INTEGER(A-O,S-Z)
        C
0003          DIMENSION BUF(40)
0004          DIMENSION INCHS(5),OUTCHS(5)
        C
0005          DATA UTY /6/
0006          DATA NUMCH /5/
0007          DATA Y,N,X /'Y','N','X'/
        C
        CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        C
0008          DO 120 I=1,NUMCH
0009             INCHS(I) = 0
0010             OUTCHS(I) = 0
0011      120 CONTINUE
        C
0012          WRITE(UTY,4100)
0013     4100 FORMAT(/'$PROCESS CHANNELS 1,2,3,4? (Y/N) >')
0014      140 READ(UTY,4101)ALPHA
0015     4101 FORMAT(20A1)
        C
0016          IF(ALPHA.EQ.X) CALL EXIT
0018          IF(ALPHA.EQ.Y) GO TO 141
0020                         GO TO 146
        C     THEN                 PROCESS DEFAULT CHANNELS
0021      141    DO 145 I = 1,5
0022                CH = I
0023                IF(CH.EQ.5) GO TO 145
0025                INCHS(I) = CH
0026                OUTCHS(I) = CH
0027      145    CONTINUE
0028             GO TO 150
0029      146    CONTINUE
        C     ELSE                    SELECT INDIVIDUAL CHANNELS
0030             DO 149 I = 1,5
0031      147       WRITE(UTY,4110) I
0032     4110       FORMAT('$PROCESS CHANNEL ',I1,' TO  CHANNEL ? >')
0033                READ(UTY,4111)BUF
0034     4111       FORMAT(40A2)
0035                IF(BUF(1).EQ.X) CALL EXIT
0037                P=0
0038                CALL INTFF(P,BUF,80,OUTCH)
0039                IF(OUTCH.EQ.0) GO TO 149
0041                IF((OUTCH.GT.5).OR.(OUTCH.LT.0)) GO TO 147
0043                INCHS(I) = I
0044                OUTCHS(I) = OUTCH
0045      149     CONTINUE
        C     END IF
0046      150 CONTINUE
0047          RETURN
0048          END
```

```
      SUBROUTINE NNWNDO
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SCALES AND LOADS A WINDOW VIA NEAREST NEIGHBOR INTERPOLATION
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC TFW 11/80
C
      IMPLICIT INTEGER (A-Q,S-Z)
C
      REAL FLOAT
C
      DIMENSION DX(512), DY(512)
C
      BYTE IB(512), OB(512), ZEROS(512)
      BYTE TAB(256)
C
      LOGICAL INSERT
C
      COMMON /WINDO/ IX1,IY1,IX2,IY2,OX1,OY1,OX2,OY2,RMAG,INSERT
      COMMON /CHANL/ INCHS(5),OUCHS(5)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SET UP MAPPING FROM INPUT TO OUTPUT COORDINATES
C
      RXINC=FLOAT(IX2-IX1+1)/FLOAT(OX2-OX1+1)
      RYINC=FLOAT(IY2-IY1+1)/FLOAT(OY2-OY1+1)
C
      OY11=OY1+0
      OY21=OY2+0
      K=0
      IF(RYINC.LT.1.0) K=-1.0/(RYINC*2.0)
      L=IY1
      IF(RYINC.GT.1.0) L=FLOAT(L)+RYINC/2.0
C
      DO 20 J=OY11,OY21
         DY(J)=RYINC*K+L+0.5
         K=K+1
   20 CONTINUE
C
      OX11=OX1+1
      OX21=OX2+1
      K=0
      IF(RXINC.LT.1.0) K=-1.0/(RXINC*2.0)
      L=IX1
      IF(RXINC.GT.1.0) L=FLOAT(L)+RXINC/2.0
C
      DO 30 J=OX11,OX21
         DX(J)=RXINC*K+L+0.5
         K=K+1
   30 CONTINUE
C
      LENGTH=OX2-OX1
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     LOOP THRU THE SELECTED BANDS
C
```

compute mapping tables

```
FORTRAN IV      V01C-03          THU 12-NOV-81 14:09:30           PAGE 002
CORE=11K, UIC=[127,4]                         DB:BGWNDO,LP:=DB:BGWNDO

          DO 500 BPTR = 1,5
             BNDI = INCHS(BPTR)                                    Image loop
             BNDO = OUCHS(BPTR)
      C
      C      SKIP UNWANTED BANDS
      C
             IF(BNDI.EQ.0) GO TO 450
      C        , ELSE PROCESS LINES
      C
      C         SET UP TABLE LOOKUP
      C
                MULTIP=1
                DO 205 J=0,255
                   TEMP=MULTIP*J
                   TEMP=MIN0(TEMP,255)
                   TAB(J+1)=TEMP
       205      CONTINUE
      C
                IF(INSERT) GO TO 211                               clear memory above window
      C            ELSE ERASE LINES ABOVE OUTPUT WINDOW
      C
                   DO 210 LINE = 0,OY11-1
                      CALL IWV(BNDO,LINE,ZEROS)
       210         CONTINUE
      C         END IF
      C
       211      CONTINUE
                LASTDY = -1
                DO 400 LINE=OY11,OY21
                   NEWDY = DY(LINE)
      C
      C            SKIP PROCESSING FOR DUPLICATED LINES
      C
                   IF((NEWDY.EQ.LASTDY).AND.(.NOT.INSERT))GO TO 350
      C               ELSE PROCESS NEW OUTPUT LINE
      C
                      CALL IRV(BNDI,NEWDY,IB)                      perform nearest
      C                                                            neighbor resampling
      C               MOVE DATA TO OUTPUT BUFFER
      C
                      IF (INSERT) CALL IRV(BNDO,LINE,OB)
                      CALL LNSCAL(IB,TAB,OB,DX,OX1,LENGTH)
      C            END IF
       350         CONTINUE
      C
      C            WRITE OUTPUT LINE TO IAC
      C
                   CALL IWV(BNDO,LINE,OB)
                   LASTDY = NEWDY
       400      CONTINUE
      C
                IF(INSERT) GO TO 411
      C            ELSE ERASE LINES BELOW OUTPUT WINDOW
      C
                   DO 410 LINE = OY21+1,511                        clear memory below window
                      CALL IWV(BNDO,LINE,ZEROS)
       410         CONTINUE
      C         END IF
       411      CONTINUE
```

```
      C           END IF
0073    450       CONTINUE
0074    500    CONTINUE
      C
0075            RETURN
      C
0076           END
```

```
FORTRAN IV      V01C-03          THU 12-NOV-81 14:09:46          PAGE 001
CORE=11K, UIC=[127,4]                   DB:BGWNDO,LP:=DB:BGWNDO

      SUBROUTINE AVWNDO
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SCALES AND LOADS A WINDOW BY AVERAGING LOCAL AREAS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC TFW 11/80
C
      IMPLICIT INTEGER (A-Q,S-Z)
C
      REAL FLOAT
C
      DIMENSION DX(512), DY(512)
      DIMENSION SUMB(512)
C
      BYTE OUTB(512), ZEROS(514)
      BYTE ROWSB(512)
      BYTE TAB(256)
      BYTE NEWB(512), OLDB(512)
C
      LOGICAL INSERT
C
      EQUIVALENCE (ROWSB(2),ZEROS(514))
C
C
      COMMON /WINDO/ IX1,IY1,IX2,IY2,OX1,OY1,OX2,OY2,RMAG,INSERT
      COMMON /CHANL/ INCHS(5),OUCHS(5)
C
      DATA MINLIN/4/, MAXLIN/509/
      DATA NUMPIX/512/
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SET UP MAPPING FROM INPUT TO OUTPUT COORDINATES
C
      RXINC=FLOAT(IX2-IX1+1)/FLOAT(OX2-OX1+1)
      RYINC=FLOAT(IY2-IY1+1)/FLOAT(OY2-OY1+1)
C
      OY11=OY1+0
      OY21=OY2+0
      K=0
      L=IY1
C
      DO 20 J=OY11,OY21
          DY(J)=RYINC*K+L+0.5
          K=K+1
   20 CONTINUE
C
      OX11=OX1+1
      OX21=OX2+1
      K=0
      L=IX1
C
      DO 30 J=OX11,OX21
          DX(J)=RXINC*K+L+0.5
          K=K+1
   30 CONTINUE
C
C     SET OUTPUT LENGTH, AND NUMBER OF COLUMNS & ROWS TO BE AVERAGED
```

compute mapping tables

```
FORTRAN IV      V01C-03         THU 12-NOV-81 14:09:46           PAGE 002
CORE=11K, UIC=[127,4]                        DB:BGWNDO,LP:=DB:BGWNDO
        C
0034          LEN=OX2-OX1
0035          NCOL=1.0/RMAG
0036          NROW=NCOL
0037          AREA=NCOL*NROW
        C
0038          MULTIP=1
0039          DO 41 J=0,255
0040              TEMP=MULTIP*J
0041              TEMP=MIN0(TEMP,255)
0042              TAB(J)=TEMP
0043     41   CONTINUE
        C
        CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        C
        C     LOOP THRU THE SELECTED BANDS
        C
0044          DO 500 BPTR = 1,5
0045              BNDI = INCHS(BPTR)
0046              BNDO = OUCHS(BPTR)
        C
        C         SKIP UNWANTED BANDS
        C
0047              IF(BNDI.EQ.0) GO TO 450
        C             ELSE PROCESS LINES
        C
0049                  IF(INSERT) GO TO 211
        C                 ELSE ERASE LINES ABOVE OUTPUT WINDOW
        C
0051                      DO 210 LINE = 0,OY11-1
0052                          CALL IWV(BNDO,LINE,ZEROS)
0053    210               CONTINUE
        C             END IF
0054    211           CONTINUE
        C
0055                  TOPAVL=DY(OY11)-NROW+NROW/2
0056                  TOPAVL=MAX0(TOPAVL,MINLIN)
        C
0057                  DO 400 LINE=OY11,OY21
0058                      NEWDY = DY(LINE)
        C
        C                 COMPUTE RUNNING COLUMN SUM ON INPUT LINES UNTIL
        C                 SUMS ARE CENTERED OVER NEXT INPUT LOCATION NEWDY
        C
0059                      BOTAVL=DY(LINE)+NROW/2
        C
0060                      DO 390 NEWLNI=TOPAVL,BOTAVL
        C
        C                     COMPUTE LINE LIMITS OF INPUT AREA
        C
0061                          OLDLNI=NEWLNI-NROW
0062                          OLDLNI=MAX0(OLDLNI,MINLIN)
0063                          OLDLNI=MIN0(OLDLNI,MAXLIN)
0064                          NEWLNI=MIN0(NEWLNI,MAXLIN)
        C
        C                     UPDATE RUNNING COLUMN SUMS IN SUMB
        C
0065                          COUNT=COUNT+1
0066                          IF(COUNT.GT.NROW)CALL IRV(BNDI,OLDLNI,OLDB)
```

```
FORTRAN IV      V01C-03          THU 12-NOV-81 14:09:46          PAGE 003
CORE=11K, UIC=[127,4]                        DB:BGWNDO,LP:=DB:BGWNDO
0068                          CALL IRV(BNDI,NEWLNI,NEWB)
0069                          CALL COLSUM(NEWB,OLDB,SUMB,NUMPIX)
          C
          C                   SET STARTING INPUT LINE FOR FUTURE LOOP
          C
0070                          TOPAVL=BOTAVL+1
0071       390             CONTINUE
          C
          C                PROCESS OUTPUT LINE
          C
0072                       IF(INSERT) CALL IRV(BNDO,LINE,OUTB)
0074                       CALL ROWSUM(SUMB,ROWSB,NUMPIX,NCOL,AREA)
0075                       CALL LNSCAL(ROWSB,TAB,OUTB,DX,OX1,LEN)
0076                       CALL IWV(BNDO,LINE,OUTB)
0077       400          CONTINUE
          C
0078                    IF(INSERT) GO TO 411
          C                ELSE ERASE LINES BELOW OUTPUT WINDOW
          C
0080                       DO 410 LINE = OY21+1,511
0081                          CALL IWV(BNDO,LINE,ZEROS)
0082       410             CONTINUE
          C             END IF
0083       411          CONTINUE
          C          END IF
0084       450       CONTINUE
0085       500    CONTINUE
          C
0086               RETURN
          C
0087              END
```

compute column sums

compute area average
and write output

```
FORTRAN IV      V02C-03         THU 12-NOV-81 14:10:02          PAGE 001
CORE=11K, UIC=[127,4]                          DB:BGWNDO,LP:=DB:BGWNDO
      SUBROUTINE CDWNDO
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SCALES AND LOADS A WINDOW VIA CONTEXT DEPENDANT INTERPOLATION
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC TFW 11/80
C
      IMPLICIT INTEGER (A-O,S-Z)
C
      REAL FLOAT
C
      DIMENSION DX(512,2), DY(512,2)
C
      BYTE IB(512,2), OB(512), ZEROS(512)
      BYTE BYTE
C
      LOGICAL INSERT
C
      COMMON /WINDO/ IX1,IY1,IX2,IY2,OX1,OY1,OX2,OY2,PMAG,INSERT
      COMMON /CHANL/ INCHS(5),OUCHS(5)
C
      EQUIVALENCE (BYTE,WORD)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SET UP MAPPING FROM INPUT TO OUTPUT COORDINATES
C
      RXINC=FLOAT(IX2-IX1+1)/FLOAT(OX2-OX1+1)
      RYINC=FLOAT(IY2-IY1+1)/FLOAT(OY2-OY1+1)
C
      OY11=OY1+0
      OY21=OY2+0
      K=0
      IF(RYINC.LT.1.0) K=+1.0/(RYINC*2.0)
      L=IY1
C
      DO 20 J=OY11,OY21
         RPOSIT=RYINC*K+L
         INTG=INT(RPOSIT)
         RFRACT=RPOSIT-FLOAT(INTG)
         FRACT=128.0*RFRACT
         DY(J,1)=INTG
         DY(J,2)=FRACT
         K=K+1
20    CONTINUE
C
      OX11=OX1+1
      OX21=OX2+1
      K=0
      IF(RXINC.LT.1.0) K=+1.0/(RXINC*2.0)
      L=IX1
C
      DO 30 J=OX11,OX21
         RPOSIT=RXINC*K+L
         INTG=INT(RPOSIT)
         RFRACT=RPOSIT-FLOAT(INTG)
         FRACT=128.0*RFRACT
         DX(J,1)=INTG
```

Compute mapping tables

```
0040             DX(J,2)=FRACT
0041             K=K+1
0042       30    CONTINUE
          C
0043             LENGTH=OX2-OX1
          C
          CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
          C
          C      LOOP THRU THE SELECTED BANDS
          C
0044             DO 500 BPTR = 1,5
0045                 BNDI = INCHS(BPTR)
0046                 BNDO = OUCHS(BPTR)
          C
          C          SKIP UNWANTED BANDS
          C
0047                 IF(BNDI.EQ.0) GO TO 450
          C              ELSE PROCESS LINES
          C
0049                     IF(INSERT) GO TO 211
          C                  ELSE ERASE LINES ABOVE OUTPUT WINDOW
          C
0051                         DO 210 LINE = 0,OY11-1
0052                             CALL IWV(BNDO,LINE,ZEROS)
0053       210               CONTINUE
          C              END IF
0054       211           CONTINUE
          C
          C              SET UP INTERPOLATION LOOP
          C
0055                     LNPNTR=1
0056                     FSTLIN=DY(OY11,1)
0057                     PRELIN=FSTLIN-1
0058                     PRELIN=MAX0(1,PRELIN)
0059                     CALL IRV(BNDI,PRELIN,IB(1,LNPNTR))
0060                     LASTDY = -1
          C
0061                     DO 400 LINE=OY11,OY21
0062                         NEWDY = DY(LINE,1)
          C
          C                  SKIP READ IF NO NEW INPUT LINE IS REQUIRED
          C
0063                         IF(NEWDY.EQ.LASTDY) GO TO 310
          C                      ELSE UPDATE INPUT BUFFERS TO CONTAIN LINES
          C                      ABOVE AND BELOW LINE TO BE INTERPOLATED
          C
0065                             LABOVE=LNPNTR
0066                             LNPNTR=3-LNPNTR
0067                             LBELOW=LNPNTR
0068                             CALL IRV(BNDI,NEWDY,IB(1,LBELOW))
          C                  END IF
0069       310               CONTINUE
          C
          C                  PIXEL BY PIXEL INTERPOLATION
          C
0070                         IF(INSERT) CALL IRV(BNDO,LINE,OB)
          C
0072                         DO 350 PIX=OX11,OX21
          C
```

read input data

```
      C               GET 4 INPUT PIXELS SURROUNDING INTERPOLATING POINT
      C
0073                  PLEFT=DX(PIX,1)
0074                  PRIGHT=PLEFT+1
      C
0075                  BYTE=IB(PLEFT,LABOVE)
0076                  A=WORD
0077                  BYTE=IB(PRIGHT,LABOVE)
0078                  B=WORD
0079                  BYTE=IB(PLEFT,LBELOW)
0080                  C=WORD
0081                  BYTE=IB(PRIGHT,LBELOW)
0082                  D=WORD
      C
      C               GET FRACTIONAL PART OF INTERPOLATING POINT LOCATION
      C
0083                  X=DX(PIX,2)
0084                  Y=DY(LINE,2)
      C
      C               SELECT AND INTERPOLATE ON A PLANE CHOSEN BY
      C               LOCAL GRADIENT AND QUADRANT OF POINT IN ABCD
      C
0085                                                        CASE=1
0086                  IF(IABS(C-B).LT.IABS(A-D))            CASE=3
0088                  IF((CASE.EQ.1).AND.(Y.LT.X))          CASE=2
0090                  IF((CASE.EQ.3).AND.((128-Y).LT.X))    CASE=4
      C
0092                  GO TO (331,332,333,334) CASE
      C
0093   331               OB(PIX)=A+(X*(D-C)+Y*(C-A))/128
0094                                                           GO TO 340
0095   332               OB(PIX)=A+(X*(B-A)+Y*(D-B))/128
0096                                                           GO TO 340
0097   333               OB(PIX)=A+(X*(B-A)+Y*(C-A))/128
0098                                                           GO TO 340
0099   334               OB(PIX)=C+(X*(D-C)+(128-Y)*(B-D))/128
      C
      C
0100   340            END CASES
                      CONTINUE
0101   350         CONTINUE
      C
      C            WRITE OUTPUT LINE TO IAC
      C
0102               CALL IWV(BNDO,LINE,OB)
0103               LASTDY = NEWDY
0104   400      CONTINUE
      C
0105            IF(INSERT) GO TO 411
      C            ELSE ERASE LINES BELOW OUTPUT WINDOW
      C
0107               DO 410 LINE = DY2+1,511
0108                  CALL IWV(BNDO,LINE,ZEROS)
0109   410         CONTINUE
      C         END IF
0110   411      CONTINUE
      C      END IF
0111   450   CONTINUE
0112   500 CONTINUE
      C
```

compute interpolated
output pixel

write output values

```
0113        RETURN
      C
0114        END
```

561

APPENDIX B

THEME FILTERING PROGRAM

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C                    SUPERVISED THEME FILTER
C
C   THIS PROGRAM PREFORMS OPERATOR CONTROLLED LOW PASS SPATIAL FILTERING
C   ON A CATEGORIZED THEMATIC IMAGE.  IT TABULATES INDIVIDUAL THEME
C   OCCURANCES OVER A RECTANGULAR FILTER AREA, AND IN ACCORDANCE WITH
C   OPERATOR SPECIFIED CONSTRAINTS, REPLACES THE CENTER PIXEL OF
C   THE RECTANGULAR AREA WITH THE LOCALLY MOST COMMON THEME.
C
C   THE PROGRAM IS PRIMARILY DESIGNED TO TRANSFORM THEMATIC IMAGES
C   PRODUCED BY SUPERVISED OR UNSUPERVISED COMPUTER CLASSIFICATION
C   TECHNIQUES INTO SPATIALLY SIMPLIFIED THEMATIC MAPS, WHICH CLOSELY
C   RESEMBLE THOSE WHICH MIGHT BE GENERATED BY MANUAL PHOTO INTERPRETATION.
C   IT PERFORMES THE SPATIAL SIMPLIFICATION NECESSARY TO CHANGE "NOISY"
C   COMPUTER THEMATIC MAPS INTO ONES WHICH CAN BE PRINTED OR STORED
C   AS OUTLINES OR POLYGONS.
C
C   BY JUDICIOUS USE OF THEME REPLACEMENT CONTRAINTS, IT CAN ALSO BE USED
C   TO RESOLVE CLASSIFICATION ERROR ON A SPATIAL BASIS, TO FILTER ONLY
C   SPECIFIC THEMES, OR TO TOTALLY REMOVE UNDESIRED FEATURES SUCH AS
C   SHADOWS.
C
C
C   INPUT:    A CLASSIFIED IMAGE WHERE EACH THEME HAS BEEN MAPPED TO
C             A SPECIFIC GREY LEVEL.
C
C             USER SPECIFIED FILTER SIZE.
C
C             USER SPECIFIED CONSTRAINTS FOR EACH THEME
C
C   OUTPUT:   SMOOTHED IMAGE MAPPED TO THE SAME GREY LEVELS
C
C
C   FORTRAN PROGRAM MODULES:
C
C      THMFIL ---MAIN PROGRAM
C
C      THRHLD ---OPERATOR INTERFACE WHICH PROMPTS FOR AND ACCEPTS
C                A SOMEWHAT LENGTHY LIST OF AREA THRESHOLDS WHICH
C                CONTROL FILTERING ACTION
C
C
C   PDP-11 ASSEMBLER MODULES:
C
C      THMTOT ---COMPUTES A SET OF RUNNING SUMS, ONE FOR EACH IMAGE COLUMN
C
C      ROWTOT ---COMPUTES A LINE OF AREAL SUMS BY TAKING A RUNNING SUM
C                ON THE COLUMN SUMS
C
C      REPLAC ---CHECKS THEME AREA TOTALS FOR EACH FILTER POSITION, CHECKS
C                CONSTRAINTS AND REPLACES CENTRAL PIXEL
C
C
C   IMAGE 100 LIBRARY SUBROUTINES CALLED:
C
C      FRONT
```

```
      C       INTFF  ---FREE FORMAT INPUT ROUTINES
      C
      C       IRC    ---READS BOX CURSOR POSITION
      C
      C       IRV
      C       IWV    ---READ AND WRITE LINES TO VIDEO MEMORY
      C
      C       OUTPUT ---TERMINAL CONTROL ROUTINE
      C
      C       TALOOK ---TABLE LOOKUP FOR VIDEO VALUES
      C
      C
      C    KEY VARIABLES:
      C
      C       BUF    ---ALPHANUMERIC INPUT STRING
      C
      C       THMVAL ---ARRAY CONTAINING GREY LEVEL OF EACH THEME
      C
      C       GROPCT ---ARRAY OF AREA PERCENTILES, ONE FOR EACH THEME. FOR A
      C                 GIVEN THEME TO GROW OR REPLACE ANOTHER, THE THEME MUST
      C                 COVER AT LEAST THIS PERCENT OF THE FILTER RECTANGLE
      C
      C       SHRPCT ---AN ARRAY OF AREA PERCENTILES WHICH CAN NOT BE EXCEEDED
      C                 IF A GIVEN THEME IS TO SHRINK OR BE REPLACED
      C
      C       ENTAB  ---LOOKUP TABLE TO MAP THEMES TO LOWEST GREY LEVELS
      C
      C       DETAB  ---LOOKUP TABLE TO MAP PROCESSED IMAGE BACK TO THEME LEVELS
      C
      C       COLSUM(PIXELS,THEMES) ---ARRAY TO STORE RUNNING COLUMN SUMS
      C
      C       THMSUM(PIXELS,THEMES) ---ARRAY TO STORE RUNNING AREA SUMS
      C
      C       NEWB
      C       MIDB
      C       OLDB   ---BUFFERS FOR BOTTOM, MIDDLE AND TOP LINES OF FILTER POSITION
      C
      C       OX1,OY1
      C       OX2,OY2 ---UPPER LEFT AND LOWER RIGHT CORNERS OF IMAGE
      C                  SEGMENT TO BE PROCESSED
      C
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
0001        SUBROUTINE THMFIL
      C
      C
      C
      CCCCCCCCCCCCCCCCCCCCCCC GENERAL * ELECTRIC CCCCCCCCCCCCCCCCCCCCCCC
      C
      C     PERFORMES SPATIAL THEME FILTERING WITH A RECTANGULAR WINDOW
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC TFW 02/81
      C
0002        IMPLICIT INTEGER (A-Q,S-Z)
0003        INTEGER R
      C
0004        DIMENSION BUF(40)
```

887

```
0005          DIMENSION THMVAL(9)
       C
0006          DIMENSION BUFFER(256,16)
       C
0007          DIMENSION CS(4608), COLSUM(512,9)
0008          EQUIVALENCE (BUFFER(1,1),CS(1),COLSUM(1,1))
       C
0009          BYTE TS(4608), THMSUM(512,9)
0010          EQUIVALENCE (TS(1),THMSUM(1,1))
       C
0011          BYTE ENTAB(256), DETAB(256), PRETAB(256), ZERO(256)
       C
0012          BYTE LINES(1536)
0013          BYTE NEWB(512),MIDB(512),OLDB(512)
0014          EQUIVALENCE (NEWB(1),LINES(1))
0015          EQUIVALENCE (MIDB(1),LINES(513))
0016          EQUIVALENCE (OLDB(1),LINES(1025))
       C
       CCCCCCCCCCCCCCCCCCCCCCCCC I-100 GLOBAL COMMON CCCCCCCCCCCCCCCCCCCCCCC
0017          COMMON/EXEC/ SW(8),BUTTON,EXUIC,AGOTO,RETTSK,REXTSK,RAUX,RCTSK
            1,             RCAUX,SDATE(5),STIME(4),PQ,GPT(17)
            2,             EXSPAR(16)
0018          COMMON/TR1D/ DCODE,LIM(3,4),MASK,CURSOR(5),CTR(18),TOTAL(4)
            1,             PAREA,RMEAN(4),RVAR(4)
            2,             TR1SPR(5)
0019          COMMON/TRND/ FILESW,NDFS(15),LREC,NREC,IERR1,NDX1,IERR2,NDX2
            1,             IERR3,NDX,DFFS(15),THRESH,FL
            2,             TRNSPR(23)
0020          COMMON/SCENE/SID(5), SF(4),ERTSC(2,2),IMAGC(2,2),RCF,RCPLAT,RCPLNG
            1,             AA(4),SCNSPR(5)
0021          COMMON/PARAM/NLAMDA,NC,IRES(4),HSIG(2,4),HSF(2,4)
            1,             PSMSPR(10)
       CCCCCCCCCCCCCCCCCCCCCCCCC END I-100 GLOBAL COMMON CCCCCCCCCCCCCCCCCCCC
       C
       C        DEFAULT PERCENTILES FOR THEME GROWING AND SHRINKING
0022          DIMENSION GROPCT(9), SHRPCT(9), GROCNT(9), SHRCNT(9)
0023          DATA GROPCT /100, 000, 000, 600, 000, 000, 000, 000, 000, /
0024          DATA SHRPCT /100, 100, 100, 100, 100, 100, 100, 100, 100/
       C
       C        IMAGE LIMITS
0025          DATA MINLIN/4/, MAXLIN/509/, MINPIX/0/, MAXPIX/511/
       C
       C        PROCESSING BUFFER SIZE
0026          DATA BUFLEN /512/
       C
       C        THEME INFORMATION
0027          DATA THMVAL/0,1,2,4,8,16,32,64,128/
0028          DATA MAXTHM/8/
       C
0029          DATA X/'X'/, R/'R'/, C/'C'/, NUL/' '/
0030          DATA UTY /6/
       C
0031          DATA NCOL /3/, NROW/3/
       C
       CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       C
       C      LIST INSTRUCTIONS
       C
0032            CALL OUTPUT(27,12,7)
```

B3

```
 81     WRITE(UTY,8100)
 8100   FORMAT(10X,'SUPERVISED THEME FILTERING'//,' NOTE:',/,
      &        '   ...CONFLICT IS MAPPED TO HIGHEST # THEME.',/,
      &        '   ...THEME #0 REFERS TO UNCLASSIFIED PIXELS.',/,
      &        '   ...UNPROCESSED THEMES ARE LEFT INTACT.',/,
      &        '   ...TO REPLACE A THEME, SET A NEGATIVE THRESHOLD'//)
C
C       GET OPERATOR SCOOP
C
 50     WRITE(UTY,5000)
 5000   FORMAT('$ENTER SCRATCH CHANNEL  >')
        READ(UTY,1001)BUF
        IF(BUF(1).EQ.X) GO TO 900
        P=0
        CALL INTFF(P,BUF,80,BNDI)
        IF((BNDI.LT.1).OR.(BNDI.GT.4)) GO TO 50
        BNDO=5
C
C
C       SET UP LOOKUP TABLES TO ENCODE THEMES INTO LOWEST
C       NUMERICAL VIDEO LEVELS AND TO DECODE BACK TO THEMES
C
        II=0
        NXTVAL=0
        M=MAXTHM+1
C
        DO 51  I=1,256
            IF(I.GT.NXTVAL)II=II+1
            II=MIN0(II,M)
            ENTAB(I)=II-1
            NXTIND=II+1
            NXTIND=MIN0(NXTIND,M)
            NXTVAL=THMVAL(NXTIND)
C
            III=MIN0(I,M)
            DETAB(I)=THMVAL(III)
 51     CONTINUE
C
 55     CONTINUE
C
 60     WRITE(UTY,6000)
 6000   FORMAT('$ENTER NUMBER OF THEMES TO BE PROCESSED >')
        CALL OUTPUT(7)
        READ(UTY,1001)BUF
        IF(BUF(1).EQ.X) GO TO 900
        P=0
        CALL INTFF(P,BUF,80,NUMTHM)
        IF((NUMTHM.GT.MAXTHM).OR.(NUMTHM.LT.1)) GO TO 60
C
CCCCCCCCCCCCCCCCCCC START OF ITERATIVE PROCESSING CCCCCCCCCCCCCCC
C
 70     CONTINUE
C
C       STORE THEMES IN SCRATCH CHANNEL
C
        DO 75 LL=-1,510,16
            DO 73 II=1,16
                LLL=LL+II
                CALL IRV(BNDO,LLL,BUFFER(1,II))
```

issue preliminary instructions to operator

designate temporary storage space

generate look up tables to encode and decode theme levels to processing indexes

```
 73     CONTINUE
        DO 74 II=1,16
            LLL=LL+II
            CALL IWV(ENDI,LLL,BUFFER(1,II))
 74     CONTINUE
 75   CONTINUE
C
 100  CONTINUE
C
C
        WRITE(UTY,1000) NROW,NCOL
 1000   FORMAT('$FILTER SIZE = ',I3,' X',I3,'  >')
        CALL OUTPUT(7)
        READ (UTY,1001)BUF
 1001   FORMAT(40A2)
        IF(BUF(1).EQ.NUL) GO TO 125
            IF(BUF(1).EQ.X) GO TO 900
            P=0
            CALL INTFF(P,BUF,80,NCOL)
            CALL INTFF(P,BUF,80,NROW)
            IF((NCOL.GT.101).OR.(NCOL.LT.1)) GO TO 100
            IF((NROW.GT.101).OR.(NROW.LT.1)) GO TO 100
 125    CONTINUE
        AREA=NCOL*NROW
C
C       GET THEME THRESHOLDS TO CONTROL FILTERING ACTION
C
        CALL THRHLD(NCOL,NROW,NUMTHM,GROPCT,GROCNT,SHRPCT,SHRCNT)
C
C       DETERMINE PROCESSING WINDOW
C
 140    WRITE(UTY,1400)
 1400   FORMAT(/'$PROCESS (C)URSORED AREA OR (S)UBSCENE?  S>')
        CALL OUTPUT(7)
        READ (UTY,1001)BUF
        IF(BUF(1).EQ.X) GO TO 900
        IF(BUF(1).EQ.C) GO TO 145
C
C           ELSE GET LIMITS FROM I-100 COMMON
            OX1=IMAGC(1,1)+1
            OX1=MAX0(OX1,MINPIX)
            OY1=IMAGC(2,1)+1
            OY1=MAX0(OY1,MINLIN)
            OX2=IMAGC(1,2)+1
            OX2=MIN0(OX2,MAXPIX)
            OY2=IMAGC(2,2)+1
            OY2=MIN0(OY2,MAXLIN)
            GO TO 149
C
C           THEN GET LIMITS FROM CURSOR POSITION
 145        CONTINUE
            CALL IRF(CURSOR)
            OX1=CURSOR(2)-CURSOR(3)
            OX2=OX1+CURSOR(3)*2
            OY1=CURSOR(4)-CURSOR(5)
            OY2=OY1+CURSOR(5)*2
C
C       END IF
 149    CONTINUE
```

operator selection of processing parameters

selection of processing window

```
              NPIX=OX2-OX1
C
C             ZERO SUMMING BUFFERS BEFORE ITERATION
C
              DO 160 I=1,BUFLEN*(MAXTHM+1)
                  CS(I)=0
                  TS(I)=0
      160     CONTINUE
              DO 161 I=1,BUFLEN*3
                  LINES(I)=0
      161     CONTINUE
C
C
C                SET UP NEGATIVE START LOCATIONS FOR BOX
C
                 BOTSTP=OY1-(NROW+1)/2
                 TOPSTP=BOTSTP-NROW
                 LINSTP=BOTSTP-NROW/2
C
                 BOTAVL=BOTSTP
                 TOPAVL=TOPSTP
C
                 DO 399 LINE=LINSTR,OY2
C
C                    READ TOP,BOT,AND MID LINES OF BOX (WITH REFLECTION)
C
                     BLREAD=BOTAVL
                     IF(BOTAVL.LT.OY1) BLREAD=-BOTAVL+OY1+OY1-1
                     IF(BOTAVL.GT.OY2) BLREAD=-BOTAVL+OY2+OY2
                     TLREAD=TOPAVL
                     IF(TOPAVL.LT.OY1) TLREAD=-TOPAVL+OY1+OY1-1
C
                     CALL IRV(BNDI,TLREAD,OLDB)
                     CALL IRV(BNDI,LINE,MIDB)
                     CALL IRV(BNDI,BLREAD,NEWB)
C
                     IF(TOPAVL.GE.BOTSTR) GO TO 305
C                       ELSE ZERO TOP LINE
                        CALL TALOOK(OLDB,ZERO,OLDB,512)
C                    END IF
      305            CONTINUE
C
C
                     BOTAVL=BOTAVL+1
                     TOPAVL=TOPAVL+1
C
C                    ENCODE INPUT VALUES TO 0-MAXTHM
C
                     CALL TALOOK(LINES,ENTAB,LINES,1536)
C
C                    UPDATE STORED COLUMN SUMS
C
                     IF(TOPAVL.LT.BOTSTR)      GO TO 310
                                               GO TO 320
C
C                       THEN PRELOAD SUMMING BUFFERS
C
      310               CONTINUE
                        DO 315 PIX=OX1,OX2
                           B=NEWB(PIX)
```

initialize summing buffers

read theme data
from memory

```
FORTRAN IV      V01C-03          [illegible]
C  E=11K, UIC=[127,4]                              DB:THMFIL,LP:=DB:THMFIL

0163                    INDEX=PIX+BUFLEN*B
0164                    CS(INDEX)=CS(INDEX)+1
0165      315       CONTINUE
0166                                       GO TO 325
       C
       C            ELSE ADD THEMES FROM NEW LINE AND SUBTRACT
       C                 THEMES FROM OLD VIA ASSEMBLER ROUTINE
       C
0167   320          CONTINUE
       C
       CCCCCCCCCCCC   FORTRAN REPLACED    CCCCCCCCCCCCCCCCC
       C            DO 322 PIX=OX1,OX2
       C                 B=NEWB(PIX)
       CC                INDEX=PIX+BUFLEN*B
       C                 CS(INDEX)=CS(INDEX)+1
       C
       C                 T=OLDB(PIX)
       C                 INDEX=PIX+BUFLEN*T
       C                 CS(INDEX)=CS(INDEX)-1
       C322         CONTINUE
       C
       CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
0168                CALL THMTOT(NEWB,OLDB,CS,OX1,NUMPIX,BUFLEN)
       C
       C        END IF
0169    325     CONTINUE
       C
       C        COMPUTE AREAL SUMS
       C
0170            IF(LINE.LT.OY1) GO TO 395
0172                ONE=1
0173                DO 330 INDEX=1,NUMTHM+1
       C
0174                    CALL ROWTOT(COLSUM(OX1,INDEX),
          &             THMSUM(OX1,INDEX),NUMPIX,NCOL,ONE)
0175    330         CONTINUE
       C
       CCCCCCCCCC FOLLOWING CODE EXECUTED BY ASSEMBLER ROUTINE REPLAC CCCCCCCCC
       CC
       CCC                REPLACE OUTPUT WITH MOST HEAVILY WEIGHTED THEME
       CCC
       CC               DO 350 PIX=OX1,OX2
       CC                   KEYIND=MIDB(PIX)+1
       CC                   IF(THMSUM(PIX,KEYIND).GT.SHRCNT(KEYIND))
       CC    &              GO TO 348
       CCC                      ELSE PIXEL CAN BE CHANGED
       CC                       MAXTOT=0
       CC                       DO 346 INDEX=1,NUMTHM+1
       CC                           TOT=THMSUM(PIX,INDEX)
       CC                           IF(TOT.LE.MAXTOT) GO TO 344
       CC                           IF(GRUCNT(INDEX).LT.0) GO TO 344
       CC                           IF(TOT.LT.GRUCNT(INDEX))GO TO 344
       CCC                              ELSE REPLACEMENT POSSIBLE
       CC                               MAXTOT=TOT
       CC                               MIDB(PIX)=INDEX-1
       CCC                          END IF
       C344                         CONTINUE
       C346                     CONTINUE
       CCC                  END IF
```

update column sums

update area sums

execute replacement logic

```
        C348                       CONTINUE
        C350                    CONTINUE
        CC
        CCCCCCCCCCC                   END  OF REPLACEMENT          CCCCCCCCCCC
        C
 0176                           NTHEME=NUMTHM+1
 0177                           CALL REPLAC(MIDB,TS,OX1,NUMPIX,NTHEME,
            &                            GROCNT,SHRCNT,BUFLEN)
        C
        C                       DECODE AND WRITE OUTPUT LINE
        C
 0178                           CALL TALOOK(MIDB,DETAB,MIDB,512)
 0179                           CALL IWV(BNDO,LINE,MIDB)
        C
        C                    END IF
 0180    395                 CONTINUE
 0181    399              CONTINUE
        C
 0182    400              WRITE(UTY,4000)
 0183    4000             FORMAT(/'$(C)HANGE THRESHOLDS, (R)ECYCLE, OR E(X)IT >')
 0184                     CALL OUTPUT(7)
 0185                     READ(UTY,1001)BUF
 0186                     IF(BUF(1).EQ.C) GO TO 100
 0188                     IF(BUF(1).EQ.R) GO TO 70
 0190                     IF(BUF(1).EQ.X) GO TO 900
 0192                     GO TO 400
        C
        C
 0193    900   CONTINUE
 0194          RETURN
        C
 0195          END
```

write output time

```
FORTRAN IV      V01C-03          THU 19-NOV-81 21:10:57         PAGE 001
CORE=11K, UIC=[127,4]                          DB:THMFIL,LP:=DB:THMFIL
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
            SUBROUTINE THRHLD(NCOL,NROW,NUMTHM,GROPCT,GROCNT,SHRPCT,SHRCNT)
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
      C       OPERATOR INTERFACE FOR THEME FILTERING
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC TFW 2/81 CCCC
      C
            IMPLICIT INTEGER (A-Z)
      C
            REAL RAREA
      C
            DIMENSION BUF(40),DAT(6),TIM(6)
            DIMENSION SHRPCT(9), GROPCT(9)
            DIMENSION SHRCNT(9), GROCNT(9)
            DIMENSION ACTION(6,9)
      C
      C
            DATA ACTION
           & /'RE','PL','AC','E ','  ','  ','GR','OW',' O','NL','Y ','  '
           & ,'RE','MO','VE',' H','OL','FS','NO',' C','HA','NG','E ','  '
           & ,'RE','MO','VE',' I','SL','ES','CL','EA','N ','NO','IS','E '
           & ,'SM','OO','TH','  ','  ','  ','SH','RI','NK',' O','NL','Y '
           & ,'UN','CE','RT','IN','N ','  '/
      C
            DATA MAXTHM/8/
            DATA Y /'Y'/
            DATA N /'N'/
            DATA NULL /' '/
            DATA UTY /6/
      C
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C
      C     SET THEMES NOT TO BE PROCESSED FOR NO CHANGE
      C
            IF(NUMTHM.GE.MAXTHM) GO TO 90
      C         ELSE SET NO CHANGE
                DO 80 TM=NUMTHM,MAXTHM-1
                    SHRPCT(TM+2)=0
                    GROPCT(TM+2)=100
         80     CONTINUE
         90 CONTINUE
      C
      C     DISPLAY AND CHANGE THEME THRESHOLDS
      C
        100     CONTINUE
                IF(CHANGE.EQ.1)GO TO 101
                    CALL OUTPUT(.,7,12,7)
                    CALL DATE(DAT)
                    CALL TIME(TIM)
                    WRITE (UTY,1000)NROW,NCOL,DAT,TIM
       1000         FORMAT(6X,I3,...,I3,' FILTER',8X,6A2,5X,6A2)
      C
                END IF
        101     WRITE(UTY,1010)
       1010     FORMAT(/16X'   THEME THRESHOLD PERCENTILES')
```

display time and current thresholds
for operator verification

```
 110        WRITE(UTY,1100)
 1100       FORMAT(/,3X,'THEME      SHRINK IF    GROW IF     FILTERING'/
     &          ,8X,'NUMBER     BELOW, %     ABOVE, %      ACTION'/)
 C
            AREA=NROWS*NCOL
            RAREA=AREA
 102        DO 190 T=0,MAXTHM
               TH=T+1
               IF(GROPCT(TH).LT.0) GO TO 105
 C                ELSE COMPUTE THRESHOLDS
                  SHRCNT(TH)=SHRPCT(TH)*RAREA/100.0
                  GROCNT(TH)=GROPCT(TH)*RAREA/100.0
 105           CONTINUE
 C
               IF(T.GT.NUMTHM) GO TO 190
 C
 C             DETERMINE FILTERING ACTION
 C
               ISLES=0.20*RAREA
               HOLES=AREA-ISLES
               IF((SHRPCT(TH).GE.0).AND.(GROPCT(TH).GE.0)) GO TO 310
 C                ELSE SET UP FLAGS FOR REPLACEMENT OF THEME
                  SHRPCT(TH)=127
                  SHRCNT(TH)=127
                  GROPCT(TH)=-16000
                  GROCNT(TH)=-16000
                  ACT=1
                                                   GO TO 340
 310           CONTINUE
               IF(SHRCNT(TH).GE.1)                GO TO 320
                                            ACT=2
                  IF(GROCNT(TH).GE.HOLES)   ACT=3
                  IF(GROCNT(TH).GF.AREA)    ACT=4
                                                  GO TO 340
 320           CONTINUE
               IF(SHRCNT(TH).GT.ISLES)            GO TO 330
                                            ACT=7
                  IF(GROCNT(TH).GE.HOLES)   ACT=6
                  IF(GROCNT(TH).GE.AREA)    ACT=5
                                                  GO TO 340
 330           CONTINUE
                                            ACT=7
               IF(GROCNT(TH).GE.AREA)       ACT=8
 340           CONTINUE
 C
               IF(CHANGE.NE.1) WRITE(UTY,1020) T,SHRPCT(TH),GROPCT(TH),
     &            (ACTION(K,ACT),K=1,6)
 1020             FORMAT(10X,I1,9X,I3,8X,I3,8X,6A2,4X,'_______________')
 C
 103           IF(CHANGE.EQ.1) WRITE(UTY,1030) T,SHRPCT(TH),GROPCT(TH)
 1030             FORMAT('$ THEME ',I1,4X,' SHRINK, GROW PERCENTILES = '
     &                  ,I3,1X,',',I3,' > ')
 C
               IF(CHANGE.NE.1) GO TO 180
 C
 C                ELSE READ NEW THRESHOLDS
 109              READ(UTY,1090)BUF
 1090             FORMAT(40A1)
                  IF (BUF(1).EQ.NULL)GO TO 170
```

change percentiles to numeric count

compute approximate filtering action for operator verification

sense for any keyboard inputs and override defaults

```
                    P=0
                    CALL  INTFF(P,BUF,80,SHPPCT(TH))
                    CALL  INTFF(P,BUF,80,GRUPCT(TH))
      C         END IF
      170       CONTINUE
      C       END IF
      180     CONTINUE
      190   CONTINUE
            IF(CHANGE.NE.1) GO TO 193
              CHANGE=0
              GO TO 100
      193   WRITE(UTY,1930)
      1930  FORMAT(//)
      C
      195   WRITE(UTY,1950)
      1950  FORMAT('$CHANGE THEME THRESHOLDS?  (Y)ES, OR (N)O    >')
      196   READ(UTY,1960)CHANGE
      1960  FORMAT(A2)
            IF((CHANGE.NE.Y).AND.(CHANGE.NE.N)) GO TO 195
            IF(CHANGE.NE.Y) GO TO 198
      C     ELSE
              CHANGE=1
              GO TO 102
      198   CONTINUE
      C
          RETURN
          END
```

```
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;
                                ;               THREE ROUTINES FOR RAPID EXECUTION OF THEME FILTERING
                                ;
                                ;
                                                .TITLE THMSUM
                                                .GLOBL THMTOT
                                                .GLOBL REPLAC
                                                .GLOBL ROWTOT
                                ;
                                ;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                ;
                                ;               FIRST ROUTINE DOES RUNNING SUM OF THEMES INTO A
                                ;                       SET OF ACCUMULATING BUFFERS, ONE PER THEME
                                ;
                                ;
                                ;               FORTRAN CALLING SEQUENCE:
                                ;                       CALL THMSUM(NEWB,OLDB,SUMB,START,NUMPIX,BUFLEN)
                                ;
                                ;       NEWB = BYTE ARRAY CONTAINING NEXT LINE FOR AVERAGING AREA
                                ;       OLDB = BYTE ARRAY CONTAINING LINE TO BE SUBTRACTED FROM AREA
                                ;       SUMB = WORD ARRAY SUMB(BUFLEN,NTHM) CONTAINS COLUMN SUMS OF THEMES
                                ;       BUFLEN = NUMBER OF PIXELS IN THE LINES
                                ;
   000000                       THMTOT:
                                ;               UPDATE SUMB BY ADDING NEW LINE AND SUBTRACTING OLD
                                ;
   000000  016500  000002               MOV     2(R5),R0        ;NEW LINE ADDRESS
   000004  016501  000004               MOV     4(R5),R1        ;OLD LINE ADDRESS
   000010  016502  000006               MOV     6(R5),R2        ;SUM BUFFER ADDRES
   000014  017503  000010               MOV     @10(R5),R3      ;GET START PIXEL
   000020  017504  000014               MOV     @14(R5),R4      ;GET BUFFER LENGTH
   000024  006304                       ASL     R4              ;    IN WORDS
   000026  017505  000012               MOV     @12(R5),R5      ;SET LOOP COUNTER
                                ;
   000032  060300                       ADD     R3,R0           ;POINTS TO 1ST NEWB PIXEL
   000034  060301                       ADD     R3,R1           ;POINTS TO 1ST OLDB PIXEL
   000036  060302                       ADD     R3,R2
   000040  060302                       ADD     R3,R2           ;POINTS TO 1ST SUMB PIXEL
                                ;
                                ;
   000042  112103               LOOP:   MOVB    (R1)+,R3        ;GET THEME NUMBER FROM OLD LINE
   000044  070304                       MUL     R4,R3           ;R3=BUFLEN*THMNUM
   000046  060203                       ADD     R2,R3           ;TARGET ADDRESS
   000050  005313                       DEC     (R3)            ;DECREMENT THE SUM
                                ;
   000052  112003                       MOVB    (R0)+,R3        ;THEME NUMBER FROM NEW LINE
   000054  070304                       MUL     R4,R3
   000056  060203                       ADD     R2,R3           ;TARGET ADDRESS
   000060  005213                       INC     (R3)            ;INCREMENT THE SUM
                                ;
   000062  005722                       TST     (R2)+           ;INCREMENT TO SUMB(NEXTPIX,THM0)
                                ;
   000064  077512                       SOB     R5,LOOP
   000066  000207               FXIT1:  RTS     PC
```

```
                                     ;
                                     ;
                                     ;
                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                     ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                                     ;
                                     ;       2ND REPLACES THE KEY PIXEL WITH MOST COMMON LOCAL THEME
                                     ;           WHICH PASSES THE GROWING AND SHRINKING CONSTRAINTS
                                     ;
                                     ;       FORTRAN CALLING SEQUENCE:
                                     ;               CALL REPLAC(MIDB,SUMB,START,NPIX,NTHM,GRO,SHR,BUFLEN)
                                     ;
                                     ;       MIDB = BYTE ARRAY CONTAINING CENTER LINE OF BOX
                                     ;       SUMB = BYTE ARRAY SUMB(BUFLEN,NTHM) CONTAINS THEME COLUMN SUMS
                                     ;       NPIX = NUMBER OF PIXELS TO BE OUTPUT
                                     ;       NTHM = NUMBER OF THEMES TO BE PROCESSED
                                     ;       GRO = ARRAY GRO(NTHM) CONTAINS POPULATION THRESHOLDS WHICH
                                     ;               MUST BE EXCEEDED BEFORE A THEME IS ALLOWED TO GROW
                                     ;       SHR = ARRAY SHR(NTHM) CONTAINS POPULATION THRESHHOLDS WHICH
                                     ;               CANNOT BE EXCEEDED IF A THEME IS ALLOWED TO SHRINK
                                     ;       BUFLEN = LENGTH OF BUFFERS IN PIXELS
                                     ;
000070                               REPLAC:
000070  010667  000244                       MOV     SP,SAVE
000074  016567  000004  000240               MOV     4(R5),TH0SUM
000102  017567  000006  000234               MOV     @6(R5),START
000110  017567  000010  000230               MOV     @10(R5),NPIX
000116  017567  000012  000224               MOV     @12(R5),NTHM
000124  016567  000014  000220               MOV     14(R5),GROW
000132  016567  000016  000214               MOV     16(R5),SHRINK
000140  017567  000020  000210               MOV     @20(R5),BUFLEN
                                     ;WORDS  ASL     BUFLEN
000146  016505  000002                       MOV     2(R5),R5
                                     ;
000152  066705  000166                       ADD     START,R5
000156  005305                               DEC     R5                      ;INIT MIDB POINTER
000160  162767  000002  000154               SUB     #2,TH0SUM
                                     ;WORDS  ADD     START,TH0SUM
000166  066767  000152  000146               ADD     START,TH0SUM            ;INIT SUMB POINTER TO SUMB(START,THM0)
000174  010567  000160                       MOV     R5,STOP
000200  066767  000142  000152               ADD     NPIX,STOP               ;RETURN WHEN R5=STOP
                                     ;
                                     ;
000206  005205                       PXLOOP:     INC     R5                      ;UPDATE MIDB POINTER
000210  111501                                   MOVB    (R5) ,R1                ;GET KEY THEME NUMBER FROM MIDD
                                     ;
000212  016700  000136                           MOV     SHRINK,R0
000216  060100                                   ADD     R1,R0
000220  060100                                   ADD     R1,R0                   ;ADDRESS OF KEY THEME SHRINK THRESHOLD
                                     ;
                                     ;WORDS      ADD     #1,TH0SUM
000222  062767  000001  000112                   ADD     #1,TH0SUM               ;UPDATE SUMB POINTER TO SUMB(PIX,THM0)
000230  070167  000122                           MUL     BUFLEN,R1
000234  066701  000102                           ADD     TH0SUM,R1               ;ADDRESS OF KEY THEME TOTAL
                                     ;
000240  121110                                   CMPB    (R1),(R0)               ;COMPARE KEY THEME TOTAL TO SHRINK THRESHOLD
                                     ;WORDS      CMP     (R1),(R0)               ;COMPARE KEY THEME TOTAL TO SHRINK THRESHOLD
```

972

```
000242 003030                  BGE     PXTEST             ;JUMP OUT TO PRESERVE KEY THEME
                       ;
                       ;
000244 005000                  CLR     R0                 ;INIT MAXIMUM
000246 005001                  CLR     R1                 ;INIT THEME NUMBER
000250 016702 000076           MOV     GROW,R2            ;INIT GROW THRESHOLD POINTER
000254 016703 000062           MOV     THOSUM,R3          ;PUT SUMB(PIX,0) INTO WORKING REGISTER
                       ;
000260 012206          THLOOP:  MOV    (R2)+,SP           ;GET THEME GROW THRESHOLD
000262 111304                   MOVB   (R3),R4            ;GET THEME SUM
000264 042704 177400            BIC    #177400,R4
                       ;WORDS   MOV    (R3),R4
000270 066703 000062            ADD    BUFLEN,R3          ;SET SUM POINTER TO SUMB(PIX,NEXTHM)
                       ;
000274 020004                   CMP    R0,R4
000276 002006                   BGE    THTEST             ;JUMP OUT IF THEME NOT A MAXIMUM
                       ;
000300 020604                   CMP    SP,R4
000302 003004                   BGT    THTEST             ;JUMP OUT IF TOTAL BELOW GROW THRESHOLD
                       ;
000304 005706                   TST    SP
000306 002402                   BLT    THTEST             ;JUMP OUT IF THEME IS TO BE REPLACED
                       ;
000310 010400                       MOV     R4,R0             ;CHANGE MAXIMUM TO NEW MAX
000312 110115                       MOVB    R1,(R5)           ;AND OUTPUT THE THEME NUMBER
                       ;
000314 005201          THTEST:  INC    R1
000316 020167 000026            CMP    R1,NTHM
000322 002756                  BLT     THLOOP
                       ;
000324 020567 000030   PXTEST: CMP     R5,STOP
000330 002736                BLT     PXLOOP
                       ;
000332 016706 000002         MOV     [illegible]            ;RESTORE STACK POINTER
                       ;
000336 [illegible]     EXIT:  [illegible]
                       ;
```

B14

```
                                    ;      DATA NEEDED TO AVOID LINE SHORTENING
                                    ;
                                    ;      DOUBLE PRECISION ARITHMETIC TO AVOID OVERFLOW
                                    ;
                                    ;
                                         .TITLE  ROWTOT
                                         .GLOBL  ROWTOT
                                    ;
                                    ;    FORTRAN CALL:  CALL ROWTOT(IBUF,OBUF,LENGTH,AVELEN,DIVISR)
                                    ;
                                    ;    IBUF = WORD ARRAY CONTAINING INPUT DATA
                                    ;    OBUF = BYTE OUTPUT ARRAY
                                    ;    LENGTH = NUMBER OF PIXELS TO BE OUTPUT
                                    ;    AVELEN = NUMBER OF ELEMENTS TO BE AVERAGED
                                    ;    DIVISR = DIVISOR TO COMPUTE AVERAGE
                                    ;
000364 016567 000002 000432 ROWTOT:  MOV     2(R5),IBUF       ;SAVE LOCATIONS
000372 016567 000004 000426          MOV     4(R5),OBUF       ;
000400 017567 000006 000406          MOV     @6(R5),LENGTH    ;SAVE PARAMETERS
000406 017567 000010 000376          MOV     @10(R5),AVELEN   ;
000414 017567 000012 000360          MOV     @12(R5),DIVISR   ;
000422 010667 177712                 MOV     SP,SAVE
                                    ;
                                    ;
                                    ;    SET LOOP COUNTERS
                                    ;
000426 016767 000362 000352          MOV     LENGTH,STOP2
000434 066767 000354 000344          ADD     LENGTH,STOP2
000442 066767 000356 000336          ADD     IBUF,STOP2       ;POINTS TO END OF IBUF
                                    ;
000450 016767 000340 000332          MOV     LENGTH,STOP3
000456 066767 000344 000324          ADD     OBUF,STOP3       ;POINTS TO END OF OBUF
                                    ;
                                    ;    INITIALIZE POINTERS
                                    ;
000464 016700 000322                 MOV     AVELEN,R0
000470 005200                        INC     R0
000472 006200                        ASR     R0              ;THIS=2*((AVELEN+1)/2)
000474 010001                        MOV     R0,R1
000476 006300                        ASL     R0
000500 016767 000320 000310          MOV     IBUF,FORSTR
000506 160067 000304                 SUB     R0,FORSTR        ;STARTING ADDRESS FOR LEADING ELEMENT
                                    ;
000512 016767 000300 000300          MOV     FORSTR,BACSTR
000520 166767 000266 000272          SUB     AVELEN,BACSTR
000526 166767 000260 000264          SUB     AVELEN,BACSTR    ;STARTING ADDRESS FOR TRAILING ELEMENT
                                    ;
000534 016767 000266 000260          MOV     OBUF,OUTSTR
000542 160067 000254                 SUB     R0,OUTSTR        ;STARTING ADDRESS FOR OUTPUT BYTE
                                    ;
000546 016767 000252 000254          MOV     IBUF,OFFSET
000554 066767 000244 000246          ADD     IBUF,OFFSET      ;TO ACHIEVE LOW END REFLECTION, SUBTRACT
                                                              ;  ADDRESS FROM OFFSET
                                    ;
000562 016704 000230                   MOV     FORSTR,R4      ;SET THE POINTERS
000566 016706 000226                   MOV     BACSTR,SP
000572 016705 000224                   MOV     OUTSTR,R5
```

0 7 1

B15

870

```
000576  005002                          CLR     R2
000600  005003                          CLR     R3
                                ;
                                ;
                                ;       PROCESSING LOOP FOR THE FIRST SEGMENT OF THE LINE
                                ;
000602  010400                  LOOP1:  MOV     R4 ,R0          ;LEADING ELEMENT ADDRESS
000604  062704  000002                  ADD     #2,R4
                                ;
000610  020067  000210                  CMP     R0,IBUF         ;DOES IT POINT TO VALID DATA YET?
000614  003003                          BGT     FOROK
000616  005400                            NEG     R0              ;IF NOT REFLECT ADDRESS
000620  066700  000204                    ADD     OFFSET,R0
000624  061002                  FOROK:  ADD     (R0),R2         ;ADD LEADING ELEMENT INTO ACCUMULATORS
000626  005503                          ADC     R3              ;  WITH DOUBLE PRECISION
                                ;
000630  010600                          MOV     SP ,R0          ;TRAILING ELEMENT ADDRESS
000632  062706  000002                  ADD     #2,SP
                                ;
000636  020067  000154                  CMP     R0,FORSTR       ;ARE WE STILL PRELOADING ACCUMULATORS?
000642  002420                          BLT     TEST1
                                ;
000644  020067  000154                    CMP     R0,IBUF         ;INTO VALID DATA YET?
000650  003003                            BGT     BACOK
000652  005400                              NEG     R0              ;IF NOT, REFLECT ADDRESS
000654  066700  000150                      ADD     OFFSET,R0
000660  161002                  BACOK:    SUB     (R0),R2         ;SUBTRACT TRAILING ELEMENT FROM ACCUMULATORS
000662  005603                            SBC     R3              ;  WITH DOUBLE PRECISION
                                ;
000664  020567  000136                    CMP     R5,OBUF         ;IS OUTPUT POINTER INTO OBUF YET?
000670  002405                            BLT     TEST1
000672  010201                              MOV     R2,R1           ;IF YES, COMPUTE AVERAGE
000674  010300                              MOV     R3,R0
000676  071067  000100                      DIV     DIVISR,R0
000702  110015                              MOVB    R0,(R5)         ;AND PUT IT INTO OBUF
                                ;                   NO OPERATION
000704  005205                  TEST1:  INC     R5              ;INCREMENT OUTPUT POINTER
000706  020667  000112                  CMP     SP,IBUF         ;IS ALL INPUT PAST IBUF YET?
000712  003733                          BLE     LOOP1
                                ;
                                ;
                                ;       PROCESSING LOOP FOR THE MAIN SEMENT OF THE LINE
                                ;
000714  162602                  LOOP2:  SUB     (SP)+,R2        ;SUBTRACT TRAILING ELEMENT NORMALLY
0  716  005603                          SBC     R3
                                ;
000720  062402                          ADD     (R4)+,R2        ;THEN ADD LEADING ELEMENT NORMALLY
000722  005503                          ADC     R3
                                ;
000724  010201                          MOV     R2,R1           ;COMPUTE THE AVERAGE
000726  010300                          MOV     R3,R0
000730  071067  000046                  DIV     DIVISR,R0
                                ;
000734  110025                          MOVB    R0,(R5)+        ;OUTPUT THE AVERAGE
                                ;
000736  026704  000044                  CMP     STOP[illegible],R4
000742  003364                          BGT     LOOP2
```

B16

```
                                    ;
                                    ;
                                    ;       PROCESSING LOOP FOR THE LAST SEGMENT OF THE LINE
                                    ;
000744  162602                      LOOP3:  SUB     (SP)+,R2        ;SUBTRACT TRAILING ELEMENT NORMALLY
000746  005603                              SBC     R3
                                    ;
000750  064402                              ADD     -(R4),R2        ;THEN ADD LEADING ELEMENT BY REFLECTION
000752  005503                              ADC     R3
000754  010201                              MOV     R2,R1           ;COMPUTE THE AVERAGE
000756  010300                              MOV     R3,R0
000760  071067  000016                      DIV     DIVISR,R0
                                    ;
000764  110025                              MOVB    R0,(R5)+        ;OUTPUT THE AVERAGE
                                    ;
000766  026705  000016                      CMP     STOP3,R5
000772  003364                              BGT     LOOP3
                                    ;
000774  016706  177340                      MOV     SAVE,SP         ;RESTORE STACK POINTER
001000  000207                              RTS     PC
                                    ;
                                    ;
001002  000000                      DIVISR:         +0
001004  000000                      STOP1:          +0
001006  000000                      STOP2:          +0
001010  000000                      STOP3:          +0
001012  000000                      AVELEN:         +0
001014  000000                      LENGTH:         +0
001016  000000                      FORSTR:         +0
001020  000000                      PALSTR:         +0
001022  000000                      OUTSTR:         +0
001024  000000                      IBUF:           +0
001026  000000                      OBUF:           +0
001030  000000                      OFFSET:         +0
                                    ;
        000001                              .END
```

FILMED

8